前 言

近年来，一些新的研究领域如神经网络、模糊系统和进化计算等，由于它们都是模拟人类的智能行为或进化过程而发展起来的，并且具有高度并行化与智能化等特征，因而引起了人们的极大兴趣。这些新方法通过“拟物”与“仿生”，为解决某些复杂问题提供了新的方法和途径，因而能够得以迅速发展并成为人们的研究热点之一。

1992年，美国学者James C. Bezdek在论文《计算智能》中讨论了神经网络、模式识别与智能之间的关系，并将智能分为生物智能、人工智能和计算智能三个层次。Bob Marks于1993年提出的理论将神经网络、遗传算法、模糊系统、进化规划和人工生命统称为计算智能。Lotfi Zadeh则认为传统的硬件计算是强调人工智能的计算模式，而计算智能的基础是软件计算，即模糊逻辑、神经网络和进化计算。他认为CI和AI的本质区别在于使用的推理类型不同，AI使用的是易脆逻辑，而CI使用的是模糊逻辑和规则。David Fogel在1995年发表的评论中指出智能就是对环境的适应能力。但他认为计算智能高于人工智能，计算智能包含人工智能。

本书认为计算智能所涉及的研究领域主要包括以下三方面：

(1)人工神经网络(Artificial Neural Network，简称ANN)，是根据人脑的生理结构和信息处理过程，来创造人工神经网络，从而模仿人的智能。

(2)模糊系统(Fuzzy System，简称FS)，试图描述和处理人的语言和思维中存在的模糊性概念，其目的也是模仿人的智能。

(3)进化计算(Evolution Computing，简称EC)，是一种模仿生物进化过程的优化方法，也属于模仿人的智能的范畴。

可见，模仿人的智能是它们共同的奋斗目标和合作的基础。将三者统称为计算智能，因为三者实际上都是计算方法。

本书是作者在近八年来为研究生讲授神经网络、模糊系统和进化计算课程的讲稿基础上，结合作者、同事及研究生们共同的科学研究成果，并从国内外文献资料中提取最主要的理论及应用成果，加以系统组织，着重讲清基本概念、原理、模型和算法，同时介绍了一些应用实例以便读者能够熟练掌握应用方法。

全书分为四篇十五章。序篇的第一章介绍了计算智能的概念和其所涉足的研究领域及其相互之间的关系。第一篇人工神经网络中的二、三、四和五章分别讲述了人工神经网络的基本模型、前馈型神经网络、反馈型神经网络和自组织竞争神经

网络模型、结构、算法及其应用实例。第二篇的模糊系统中的六、七和八章分别讲述了模糊数学基础、模糊控制理论和模糊神经网络与神经模糊系统部分。第三篇进化计算中的九、十、十一、十二、十三、十四和十五章分别讲述了遗传算法、遗传算法的数学基础、遗传算法的实现技术、遗传算法的若干改进研究、遗传算法的收敛性、用遗传算法解决组合优化问题和其他进化算法。

在完成本书的过程中，黄岚博士、李国梁博士对书中内容的选材、组织及编写提出了许多有益的意见并作了大量工作，吉林大学出版社的责任编辑吕健波、孟亚黎、唐万新老师为本书出版做了大量工作，在此向他们表示衷心的感谢！

限于作者水平，且成书仓促，书中错误和欠妥之处在所难免，恳请读者批评指正。

作　者

2001 年 11 月

目　录

序篇　计算智能

第一篇　人工神经网络

第二篇 模糊系统

第三第　进化计算

序　篇

计算智能



第一章 绪 论

第一台电子计算机的问世，迄今已有半个世纪的历史.在这期间不仅计算机本身几经更新换代，其性能日益优越，而且计算机技术也广泛应用于社会生活的各个领域.可以说计算机的诞生和发展是20世纪科学技术最伟大的成就之一.它对推动科学、技术和社会的发展起到了难以估量的作用.

电子计算机是按冯·诺依曼(Von. Neumann)的体系结构来实现算术和逻辑运算的.现在的运算速度已可达每秒几千亿次，其结果的精确和可靠程度更是人工无法比拟的.但是，电子计算机的形象思维能力却与人脑相差甚远.在人的知觉、记忆、语言、思想与获取的心理过程中，尤其是在实时处理中，人脑这一慢速和充满噪声干扰的硬件是在进行着极其复杂的宏并行处理.人们对于十分复杂的事物可以不假思索，一目了然地予以识别，但是，即使很简单的物体让先进的电子计算机来识别却相当困难.因此，要使计算机能在交互式的自然过程中，提取简单信号，表达知识及其结构，并使其与有关知识结合起来产生智能，具有较强的形象思维能力，必须突破冯·诺依曼机的体系结构，另辟新径.

制造具有智能的计算机一直是人类的梦想.直到1956年人工智能技术的出现，人们为此已做出了巨大的努力.近年来，随着人工智能应用领域的不断拓广，传统的基于符号处理机制的人工智能方法在知识表示、处理模式信息及解决组合爆炸等方面所碰到的问题已变得越来越突出，这些困难甚至使得某些学者对人工智能提出了强烈的批判，对人工智能的可能性提出了质疑.

基于上述原因，寻求一种适于大规模并行且具有某些智能特征如自组织、自适应、自学习等的算法已成为有关学科的一个研究目标.近年来，出现了一些新的研究方向如神经网络、模糊控制和进化计算等，由于它们都是模拟某一自然现象或过程而发展起来的，并且具有高度并行化与智能化等特征，因而引起了人们的极大兴趣.这些新方法通过“拟物”与“仿生”以使问题得到解决，它们也许能为解决某些复杂问题提供新的契机.至此，计算智能应运而生.

§1.1 关于计算智能

1.1.1 什么是计算智能

计算智能(Computational Intelligence，简称CI)并不是一个新的术语，早在1988年加拿大的一种刊物便以CI为名.1992年，美国学者James C. Bezdek在论文《计算智能》中讨论了神经网络、模式识别与智能之间的关系，并将智能分为生物智能、人工智能和计算智能三个层次.1993年，Bob Marks写了一篇关于计算智能和人工智能区别的文章，并在文中给出了对CI的理解.1994年的国际计算智能会议(WCCI)的命名就部分地源于Bob的文章，这次IEEE会议将国际神经网络学会(NNC)发起的神经网络(ICNN)、模糊系统(FUZZ)和进化计算(ICEC)三个年度性会议合为一体，并出版了名为《计算智能》的论文集.此后，CI这个术语就

开始被频繁地使用,同时也出现了许多关于 CI 的解释.

1992 年, James C. Bezdek 提出, CI 是依靠生产者提供的数字、数据材料进行加工处理,而不是依赖于知识;而 AI 则必须用知识进行处理. 1994 年,James 在 Florida, Orlando, 94 IEEE WCCI 会议上再次阐述他的观点,即智能有三个层次:①生物智能(Biological Intelligence, 简称 BI),是由人脑的物理化学过程反映出来的,人脑是有机物,它是智能的基础. ②人工智能(Artificial Intelligence, 简称 AI),是非生物的,人造的,常用符号来表示,AI 的来源是人类知识的精华. ③计算智能(Computational Intelligence, 简称 CI),是由数学方法和计算机实现的,CI 的来源是数值计算的传感器. 这三个层次从复杂程度由高至低的顺序为,BI、AI、CI,CI 与 AI 的差距要比 AI 与 BI 的差距小得多;而从所属关系来看:CI 是 AI 的一个子集,而 AI 虽不是 BI 的子集,BI 通常用来指导 AI 模型,同样也指导了 CI. AI 是 CI 到 BI 的过渡,因为 AI 中除计算方法之外,还包括符号表示及数值信息处理. James 还阐述了模糊系统(Fuzzy System,简称 FS)特别适合从 CI 到 AI 的平滑过渡.

Bob Marks 于 1993 年提出的理论是神经网络、遗传算法、模糊系统、进化规划和人工生命,统称为计算智能. 他的这一定义得到许多学者的认同,包括 IEEE 的领导们,其中 NNC 的主席 Walter Karplus 在 1996 年的 ADCOM 会议上重述了这种观点,认为 CI 着重研究的是系统的工作方式,而不是高强度计算.

Lotfi Zadeh 则认为传统的硬件计算是强调人工智能的计算模式,而计算智能的基础是软件计算,即模糊逻辑、神经网络和进化计算. 他认为 CI 和 AI 的本质区别在于使用的推理类型不同,AI 使用的是易脆逻辑,而 CI 使用的是模糊逻辑和规则.

David Fogel 在 1995 年发表的评论中指出智能就是对环境的适应能力. 但他认为 CI 高于 AI ,CI 包含 AI.

Eberhart,Dobbins 和 Simpson 关于 CI 的理解是,将智能系统置于一个环境中,智能行为的标准是改变或作用于环境的能力. 而 CI 只是智能系统的一个内部节点,适应性只是 CI 的衡量指标. CI 是一种方法论,是通过计算实现适应和处理新形势的能力,具有推理的属性,能得到预测或决定的结果.

另外,也有些人认为 CI 和 AI 仅有部分重合. 他们认为,AI 是符号主义,基于知识、规则和推理,相当于人的左脑;而 CI 属于连接主义,基于数据、学习和记忆,相当于人的右脑.

1.1.2 计算智能所包含的领域

计算智能所涉及的研究领域主要包括以下三方面:

(1)人工神经网络(Artificial Neural Network, 简称 ANN),是根据人脑的生理结构和信息处理过程,来创造人工神经网络,从而模仿人的智能.

(2)模糊系统(Fuzzy System, 简称 FS),试图描述和处理人的语言和思维中存在的模糊性概念,其目的也是模仿人的智能.

(3)进化计算(Evolution Computing, 简称 EC),是一种模仿生物进化过程的优化方法,也属于模仿人的智能的范畴.

可见,模仿人的智能是它们共同的奋斗目标和合作的基础. 将三者统称为计算智能,因为三者实际上都是计算方法.

§1.2 人工神经网络

1.2.1 什么是神经网络

人工神经网络是指模拟人脑神经系统的结构和功能，运用大量的处理部件，由人工方式构造的网络系统.

神经网络理论突破了传统的、串行处理的数字电子计算机的局限，是一个非线性动力学系统，并以分布式存储和并行协同处理为特色，虽然单个神经元的结构和功能极其简单有限，但是大量的神经元构成的网络系统所实现的行为却是极其丰富多彩的.

1.2.2 人工神经网络研究的历史

神经网络的研究已有 50 多年的历史，但其发展是不平衡的，它的兴衰还与"人工智能走什么路"这一争议问题有关. 由于其结构的复杂性，起始阶段进展不快，并一度陷人低谷，但仍有不少有识之士在极其艰难的条件下坚持研究，使研究工作始终没有中断，并在模型建立等理论方面取得了突破性的成果，时至今天人工神经网络成了信息领域的热门研究课题.

1. 第一阶段——初始发展期 （20 世纪 40 年代～20 世纪 60 年代）

早在 1943 年，美国心理学家 McCulloch 和数学家 Pitts 联合提出了形式神经元的数学模型，即 M-P 模型，从此开创了神经科学理论研究的新纪元. M-P 模型能够完成有限的逻辑运算. 1944 年，心理学家 Hebb 提出了改变神经元间连接强度的 Hebb 规则，它们至今仍在各种神经网络模型中起着重要的作用. 1957 年，计算机科学家 Rosenblatt 用硬件完成了最早的神经网络模型，即感知器(Perceptron)，并用来模拟生物的感知和学习能力. 1962 年，电机工程师 Windrow 和 Hoff 提出的自适应线性元件 Adaline，它是一个连续取值的线性网络，在信号处理系统中用于抵消通讯中的回波和噪声，应用十分广泛.

2. 第二阶段——低谷期 （20 世纪 60 年代末～20 世纪 70 年代末）

1969 年，人工智能之父 Minsky 和 Papert 发表的《感知器(Perdeptron)》一书指出，感知器无科学价值而言，连 XOR 逻辑分类都做不到，只能作线性划分. 由于 Minsky 在学术界的地位和影响，故其后若干年内，这一研究方向处于低潮. 另一方面，传统的冯·诺依曼电子数字计算机正处在发展的全盛时期，整个学术界都陶醉在成功的喜悦之中，从面掩盖了新型计算机的发展的必然.

尽管如此，在此期间仍然有不少有识之士不断努力，在极端艰难的条件下致力于这一研究，为神经网络研究的发展奠定了理论基础. Boston 大学的 Grossberg 和 Carpenter 提出了自适应共振理论 ART 网络. 芬兰的 Heisinki 大学的 Kohonen 提出了自组织映射网络. 日本的大坂大学的 Fukushima 提出了神经认知机网络模型. 日本东京大学的 Amari 对神经网络进行了数学理论的研究.

3. 第三阶段——兴盛期 （20 世纪 80 年代以后）

上世纪 70 年代末期，研究和试图模拟视听觉的人工智能专家首先遇到挫折，人们习以为常的知识难以教给计算机. 计算机的设计者和制造商也发现前面有不可逾越的线路微型化的物理极限，人们才开始思考冯·诺依曼机到底还能走多远. 同时，VLSI、脑科学、生物学、光学

的迅速发展也为人工神经网络的发展打下了基础.

1982 年,加州大学的物理学家 Hopfied 提出了 Hopfield 网络模型,并用电路实现. 1985 年,Rumelhart 提出了 BP 算法,把学习的结果反馈到神经网络的隐层,来改变权系矩阵,它是迄今为止最普遍的网络. Hinton 等人提出了 Boltzman 机模型. 1988 年,蔡少堂提出了细胞神经网络模型.

近年来,神经网络理论引起了美国、欧洲与日本等国科学家和企业家的巨大热情. 新的研究小组、实验室、风险公司等与日俱增,世界各国也正在组织和实施与此有关的重大研究项目. 如美国 DARPA 计划、日本 HFSP 计划、法国尤里卡计划、德国欧洲防御计划和前苏联高技术发展计划等. 1986 年 4 月,美国物理学会在 Snowbirds 召开了国际神经网络学术会议. 1987 年 6 月,IEEE 在 San Diego 召开了国际神经网络学术会议,并成立了国际神经网络学会. 1988 年起,IEEE 和国际神经网络学会每年召开一次国际会议. 1990 年 3 月,IEEE 神经网络会刊问世.

神经网络理论的应用也已经渗透到各个领域,并在智能控制、模式识别、计算机视觉、自适应滤波、信号处理、非线性优化、语音识别、知识处理、传感技术与机器人等方面取得了令人鼓舞的进展. 神经网络代表一种新的主义,即连接主义,用于解决诸如知识表达、推理学习、联想记忆、乃至复杂的社会现象,如混沌,社会演变等复杂系统的统一模型,它将预示着一个新的工业的到来.



§1.3 模糊系统

1.3.1 模糊系统理论的起源和发展

模糊现象是在众多现象中普遍存在的,如"天气很热,味道很怪"等. 人的大脑能在信息不完整不精确的情况下做出判断与决策,即进行模糊信息处理. 模糊数学是用数学方法研究和处理客观存在的模糊现象的一门新兴学科. 它所研究的事物的概念本身是模糊的,即对一个对象是否符合这个概念难以确定,这种由于概念外延的模糊而造成的不确定性称为模糊性.

模糊理论诞生于 1965 年,美国控制论专家、加利福尼亚大学教授扎德(Zadeh L A)首先提出模糊集合的概念,发表了开创性论文《模糊集合论(Fuzzy sets)》. 他提出,模糊数学的核心思想就是运用数学手段,仿效人脑思维,对复杂事物进行模糊处理. 模糊数学在基础理论和实际应用等方面引起了各国学者的极大兴趣,并产生了许多有价值的应用和惊人的成果. 1973 年,扎德教授又提出模糊逻辑(Fuzzy Logic)的理论,并积极倡导将模糊理论向人工智能方向发展. 模糊逻辑的研究虽然时间还不很长,但在智能模拟和智能控制等领域却已有了飞快的发展. 1974 年,印度裔英国学者马德尼(Mamdani E H)首先将模糊理论用于锅炉和蒸汽机的控制,并实验成功,开创了模糊控制的新领域. 20 世纪 80 年代后期以来,在日本采用模糊控制技术的家电产品大量上市,模糊技术在图像识别、自动控制、市场预测、人工智能等领域普遍应用,掀起了一股模糊热. 日本、美国和我国都成功地研制出了智能化的模糊推现机,这表明了模糊理论的强大生命力和伟大意义. 另一方面,模糊理论在学术界也得到了普遍的认同和重视. 1992 年,IEEE 召开了第一届关于模糊系统的国际会议(FUZZ-IEEE),并决定以后每年举行一次. 1993 年 IEEE 创办了专刊 IEEE Transaction on Fuzzy System. 当前,模糊理论和应用正

向深度和广度进一步发展,发展的速度越来越快,研究成果大量涌现,已经成为世界各国高科技竞争的重要领域之一.

1.3.2 模糊系统的研究范畴

模糊系统基于模糊数学理论,能对复杂事物进行模糊处理.模糊数学的理论基础包括模糊逻辑、模糊规则、模糊推理、隶属度和模糊集合等.另外,以模糊数学为基础的模糊控制器和模糊神经网络的理论和设计,将在第二篇中详细讲解.

§1.4 进化计算

进化计算(Evolution Computing)是采用简单的编码技术来表示各种复杂的结构,并通过简单的遗传操作和优胜劣汰的自然选择来指导学习和确定搜索的方向.由于它采用种群(即一组表示)的方式组织搜索,这使得它可以同时搜索解空间内的多个区域,特别适合大规模并行计算.进化计算具有自组织、自适应、自学习的特点,并且不受其搜索空间限制性条件(如可微、单峰等)的约束,不需要其他辅助信息(如导数).这使得进化计算不仅能获得较高的效率,而且操作简单、通用性强.

1.4.1 进化计算的发展过程

进化计算在20世纪六七十年代并未受到普遍的重视.其主要原因,一是因为这些方法本身还不够成熟;二是由于这些方法需要较大的计算量,而当时的计算机还不够普及且速度较慢,这样便限制了它们的应用;三是当时基于符号处理的人工智能方法正处于其顶峰时期,使得人们难以认识到其他方法的有效性及适应性.到了20世纪80年代,人工智能方法的局限性越来越突出,并且随着计算机速度的提高和并行计算机的普及,已使得进化计算对机器速度的要求不再是制约其发展的因索.进化计算的不断发展及其在一些应用领域内取得的成功,已表现出了良好的应用前景.

由于进化计算在机器学习、过程控制、经济预测、工程优化等领域取得的成功,引起了各领域科学家们的极大兴趣,自20世纪80年代中期以来,世界上许多国家都掀起了进化计算的研究热潮.目前,有数种以进化计算为主题的国际会议在世界各地定期召开,并已出版了多种有关进化计算的杂志.可以预料,随着进化计算理论研究的不断深人和应用领域的不断拓广,进化计算必将取得更大的成功.

1.4.2 进化计算的主要分支

进化计算的三大分支包括:遗传算法(Genetic Algorithm ,简称 GA)、进化规划(Evolutionary Programming,简称 EP)和进化策略(Evolution Strategies ,简称 ES).这三个分支在算法实现方面具有一些细微的差别,但它们具有一个共同的特点,即都是借助生物进化的思想和原理来解决实际问题.

下面我们分别就这三个分支作以简单的介绍.

1. 遗传算法

遗传算法是一类通过模拟生物界自然选择和自然遗传机制的随机化搜索算法,由美国

Holand J 教授于 1975 年首次提出. 它是利用某种编码技术作用于称为染色体的二进制数串，其基本思想是模拟由这些串组成的种群的进化过程，通过有组织的、然而是随机的信息交换来重新组合那些适应性好的串. 遗传算法对求解问题的本身一无所知，它所需要的仅是对算法所产生的每个染色体进行评价，并根据适应性来选择染色体，使适应性好的染色体比适应性差的染色体有更多的繁殖机会. 遗传算法尤其适用于处理传统搜索方法难于解决的复杂的非线性问题，可广泛用于组合优化、机器学习、自适应控制、规划设计和人工生命等领域，是 21 世纪有关智能计算中的关键技术之一.

2. 进化策略

1964 年，由德国柏林工业大学的 Rechenberg I 等人提出. 在求解流体动力学柔性弯曲管的形状优化问题时，用传统的方法很难在优化设计中描述物体形状的参数，然而利用生物变异的思想来随机地改变参数值并获得了较好的结果. 随后，他们便对这种方法进行了深入的研究和发展，形成了进化计算的另一个分支——进化策略.

进化策略与遗传算法的不同之处在于：进化策略直接在解空间上进行操作，强调进化过程中从父体到后代行为的自适应性和多样性，强调进化过程中搜索步长的自适应性调节；而遗传算法是将原问题的解空间映射到位串空间之中，然后再施行遗传操作，它强调个体基因结构的变化对其适应度的影响.

进化策略主要用于求解数值优化问题.

3. 进化规划

进化规划的方法最初是由美国人 Fogel L J 等人在 20 世纪 60 年代提出的. 他们在人工智能的研究中发现，智能行为要具有能预测其所处环境的状态，并按照给定的目标做出适当的响应的能力. 在研究中，他们将模拟环境描述成是由有限字符集中符号组成的序列.

1.4.3 进化计算的主要特点

进化算法与传统的算法具有很多不同之处，但其最主要的特点体现在下述两个方面：

1. 智能性

进化计算的智能性包括自组织、自适应和自学习性等. 应用进化计算求解问题时，在确定了编码方案、适应值函数及遗传算子以后，算法将根据“适者生存、不适应者淘汰”的策略，利用进化过程中获得的信息自行组织搜索，从而不断地向最佳解方向逼近.

自然选择消除了传统算法设计过程中的一个最大障碍：即需要事先描述问题的全部特点，并说明针对问题的不同特点算法应采取的措施. 于是，利用进化计算的方法可以解决那些结构尚无人能理解的复杂问题.

2. 本质并行性

进化计算的本质并行性表现在两个方面：一是进化计算是内在并行的，即进化计算本身非常适合大规模并行. 二是进化计算的内含并行性，由于进化计算采用种群的方式组织搜索，从而它可以同时搜索解空间内的多个区域，并相互交流信息，这种搜索方式使得进化计算能以较少的计算获得较大的收益.

§1.5 人工神经网络、模糊系统和进化计算的相互融合

人工神经网络能够通过学习和训练获得用数据表达的知识，除了可以记忆已知的信息之外，神经网络还具有较强的概括能力和联想记忆能力. 神经网络在智能控制、模式识别、聚类分析和专家系统等方面都取得了令人鼓舞的进展. 但神经网络的推理知识表示体现在网络连接权值上，表达比较难以理解，这是它的一个缺点.

模糊系统以扎德创立的模糊集合论为基础，抓住了人类思维中的模糊性特点，以模糊推理来处理常规方法难以解决的难题，能对复杂事物进行模糊识别、模糊度量. 目前，模糊系统在自动控制、信息处理、人工智能等方面都得到了广泛的应用. 模糊系统的显著特点是能够直接地表示逻辑，适于直接的或高级的知识表达，具有较强的逻辑功能. 但是对于模糊系统来说，模糊推理虽然是一种善于表现知识的推理方法，但它没有本质的获取知识的能力，模糊规则的确定也比较困难，通常需要领域专家知识的指导. 因此如何构造可自动处理模糊信息的模糊系统，即实现模糊规则的自动提取和模糊变量隶属度函数的自动生成及优化，一直是困扰模糊信息处理技术进一步推广的难题.

随着对模糊系统和神经网络研究的深人，两个领域间相互独立的关系逐渐改变. 如果将它们进行综合，即将符号逻辑推理方法与联接机制方法进行结合，将数值方法和模糊逻辑方法进行结合，其优势将远远高于单项研究. 模糊系统和神经网络的融合导致了模糊神经网络的产生，许多学者已对此进行了尝试.

进化计算模拟生物进化的过程，依据适者生存、优胜劣汰的进化规则，对包含可能解的种群反复进行基于遗传的操作，不断生成新的种群并使种群不断进化，同时以全局并行搜索方式来搜索优化种群中的最优个体，以求得满足要求的最优解. 其主要优点是简单、鲁棒性强、搜索空间大.

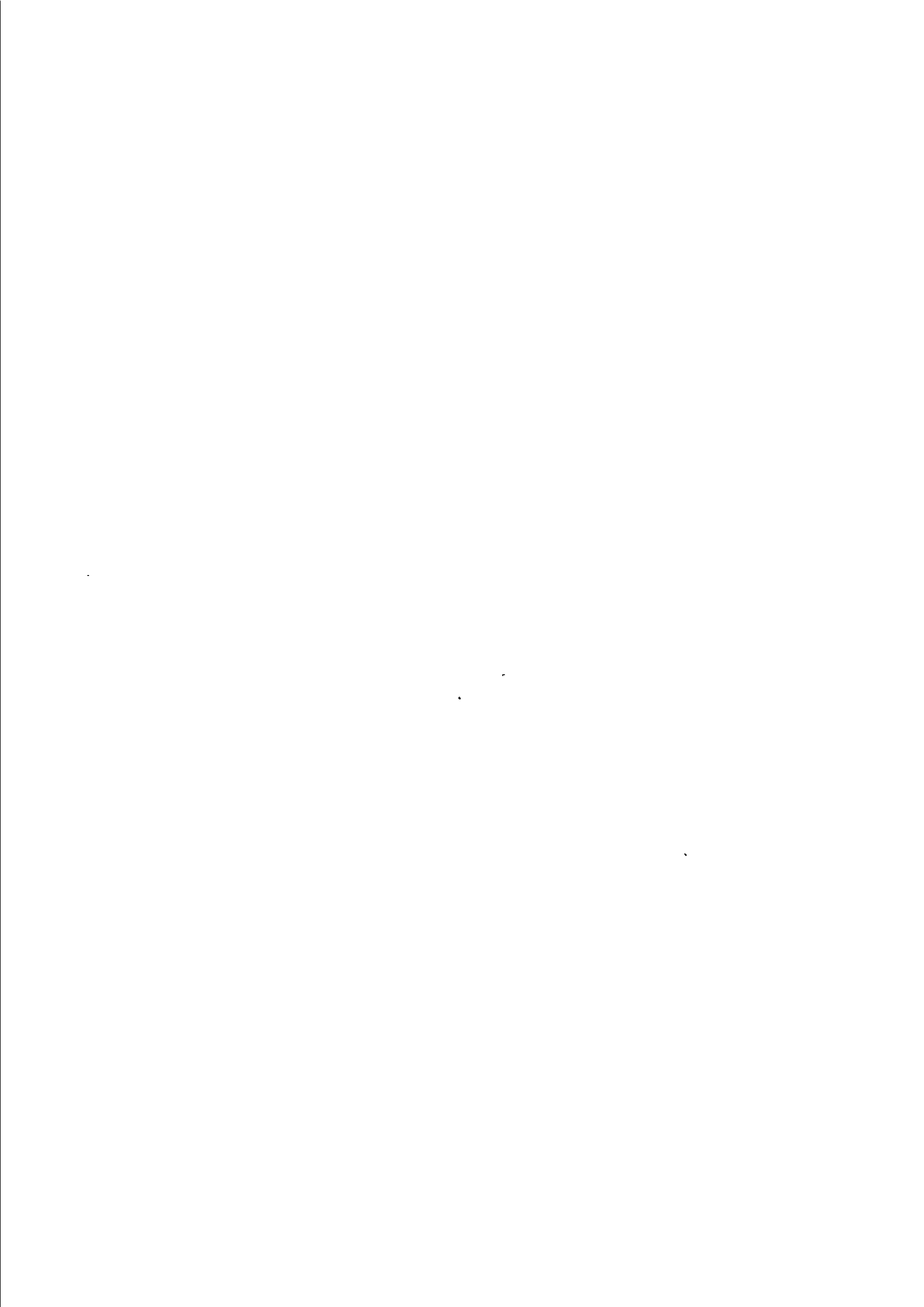
神经网络的网络结构的设计和权值的训练是一个十分重要而困难的问题，传统的方法多是凭经验或启发知识来设计网络，用梯度法来确定其中的权值，常常需要进行反复试验而且还很难找到最优的网络结构和权值. 而进化计算为神经网络的自动设计和训练提供了一种新的途径，这就是进化神经网络.

神经网络、模糊系统、进化算法三者研究同步发展、相互渗透、界限日益模糊，人们逐渐认识到由三者交叉组成的新系统具有更强的功能，因而三者相互融合的研究是当今三个分支研究的热点.



第一篇

人工神经网络



第二章　人工神经网络的基本模型

人工神经网络(ANN)是由大量的神经元互连而成的网络. 它是在现代神经科学研究成果的基础上提出的,反映了人脑功能的基本特性. 但它并不是人脑的真实描写,只是它的某种抽象、简化与模拟. 根据神经细胞的结构和功能,从 20 世纪 40 年代开始,先后提出的神经元模型有几百种之多.

§2.1　生物神经元

生物神经元是脑组织的基本单元,大约 140 亿个,尽管在种类(50 余种)、大小形状上有差异,但是作为信息处理单元,其结构和功能上大体雷同.

2.1.1　神经元的结构

本体:细脑体(细胞膜、质、核),对输入信号进行处理,相当于 CPU.

树突:本体向外伸出的分支,多根,长 1 mm 左右,本体的输入端.

轴突:本体向外伸出的最长的分支,即神经纤维,一根,长 1 cm～1 m 左右,通过轴突上的神经末梢将信号传给其他神经元,相当于本体的输出端.

突触:各神经元之间轴突和树突之间的接口,即神经末梢与树突相接触的交界面,每个细脑体大约有 10^3～10^4 个突触. 突触有兴奋型和抑制型两种.

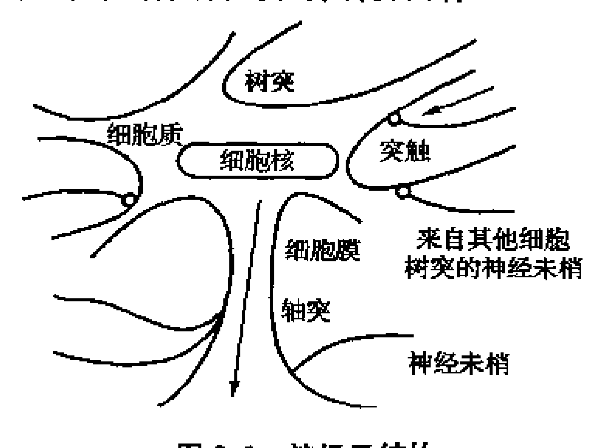


图 2.1　神经元结构

2.1.2　膜电位与神经元的兴奋

膜电位:细胞膜将细脑分为内外两部分,当外部电位为 0 时,称内部电位为膜电位.

静止脑电位:当没有输入信号时的膜电位称为静止脑电位,通常为 −70 mV 左右.

兴奋状态:当外部有输入信号时,将使膜电位发生变化,倘若使膜电位升高,比静止膜电位高 15 mV 以上,即超过 −55 mV(阈值),神经元被激活,内部电位急剧上升至 100 mV 左右,并

维持约 1 ms,然后急剧下降.相当输出一个 100 mV 高 1 ms 宽的脉冲,并沿轴突以 100 m/s 的速度传至其他的神经元.

抑制状态:当外部输入信号使膜电位下降低于阈值电位时,神经元处于抑制状态,无脉冲输出.

"兴奋—抑制"状态满足"0—1"律.

A/D 转换:电脉冲到达各突触接口后,放出某种化学物质,该物质作用于各个和其相连的神经元的细胞膜,并使其膜电位发生变化,完成了将离散的脉冲信号转换为连续变化的电位信号.

不应期:神经元输出一个脉冲后,一段时间内对激励不响应,称之为不应期,一般为几毫秒.

时间加算功能:对于不同时间通过同一突触传入的信号具有时间的加算功能.

空间加算功能:对于同一时间通过不同突触的输入信号具有空间加算功能.

§2.2 人工神经元的形式化模型

自从 1943 年,美国心理学家 McCulloch 和数学家 Pitts 合作提出了形式神经元的数学模型,即 M-P 模型以来,已有几百种神经元模型被提出.

2.2.1 M-P 模型

M-P 模型的简化图见图 2.2,其中图中的圆表示神经元的细胞体,输入线表示神经元的树突,带箭头的连接表示兴奋性突触连接,带小圆的连接表示抑制性突触的连接,输出线表示神经元的轴突,圆中的 θ 表示神经元兴奋时的阈值、与图 2.1 对照,可以看出,M-P 模型在一定程度上反映了生物神经元在结构上和功能上的特征.但是 M-P 模型给抑制性输入赋予了"否决权",只有当不存在抑制性输入而兴奋输入的加算和超过了阈值时,神经元才会兴奋.具体过程可以描述为:

$$\begin{array}{lcl}
\text{输入条件} & & \text{输出} \\
\sum_{i=1}^{N} I_i - \theta \geqslant 0 \quad \text{且} \quad \sum_{j=1}^{M} E_j = 0 \quad \text{时} & & Y = 1 \\
\sum_{i=1}^{N} I_i - \theta \geqslant 0 \quad \text{且} \quad \sum_{j=1}^{M} E_j > 0 \quad \text{时} & & Y = 0 \\
\sum_{i=1}^{N} I_i - \theta < 0 \quad \text{且} \quad \sum_{j=1}^{M} E_j = 0 \quad \text{时} & & Y = 0 \\
\sum_{i=1}^{N} I_i - \theta < 0 \quad \text{且} \quad \sum_{j=1}^{M} E_j > 0 \quad \text{时} & & Y = 0
\end{array} \qquad (2.2.1)$$

M-P 模型是神经元的一种数理逻辑(形式逻辑).因此,可以利用它来实现常规的 Boole 逻辑运算.

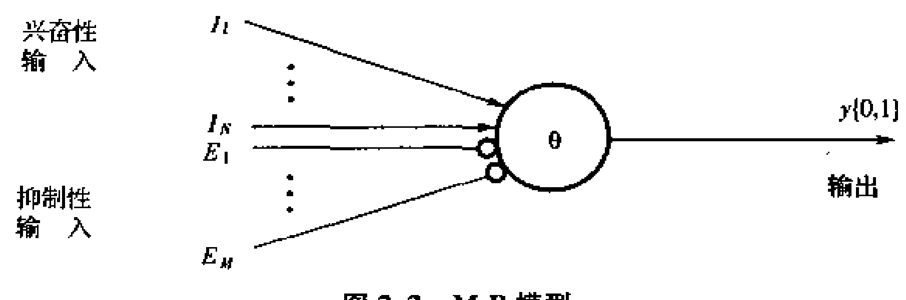
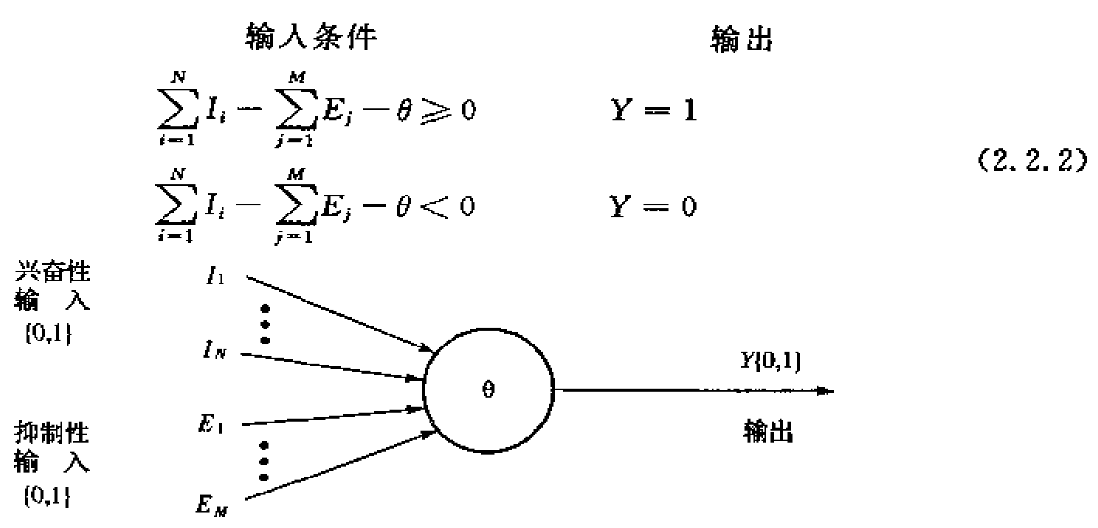


图 2.2　M-P 模型

2.2.2　线性加权模型

在 M-P 模型基础之上的一种变型，称为线性加权神经元模型，它的结构和功能如图 2.3 所示.



$$\sum_{i=1}^{N} I_i - \sum_{j=1}^{M} E_j - \theta \geqslant 0 \qquad Y = 1$$
$$\sum_{i=1}^{N} I_i - \sum_{j=1}^{M} E_j - \theta < 0 \qquad Y = 0 \tag{2.2.2}$$

图 2.3　线性加权 M-P 模型

由图看出，线性加权模型与 M-P 模型的主要区别在于对抑制性突触的联接强度的赋值. 在 M-P 模型中这个权值可以理解为 $-\infty$，在线性加权模型中为 -1.

2.2.3　阈值逻辑模型

在实际中更为常用的 M-P 模型是在前述模型的基础上的进一步改进，用可在 $[-1,1]$ 区间取值的权重来表示输入的突触连接强度，用权重的正负来表示输入突触是兴奋性突触还是抑制性突触（见图 2.4）.

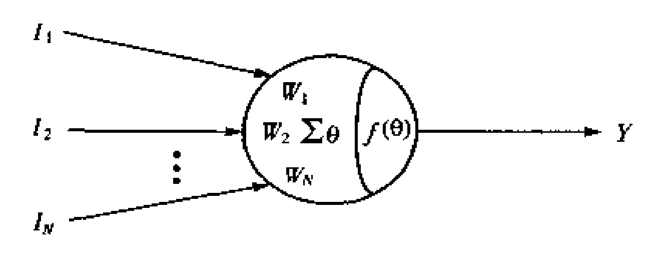


图 2.4　阈值逻辑 M-P 模型

其中：$I_i \in \{-1, +1\}$，$Y \in \{-1, +1\}$，$W_i \in [-1, +1]$，

$$Y = \mathrm{sgn}\left(\sum_{i=1}^{N} W_i I_i - \theta\right) \tag{2.2.3}$$

$$\mathrm{sgn}(x)=\begin{cases}1 & x\geqslant 0\\ -1 & x<0\end{cases}$$

M-P 模型可实现布尔逻辑运算、存储元件、模拟条件反射的建立和消退的功能.

如果把阈值也看作为一个权值,则(2.2.3)可以改写为:

$$Y=\mathrm{sgn}\left(\sum_{i=1}^{N}W_iI_i\right) \tag{2.2.4}$$

这里 $W_0=-\theta, I_0=1$.

图 2.5、图 2.6、图 2.7、图 2.8、图 2.9 分别示出了用 M-P 模型实现二元 Boole 逻辑、三元 Boole 小项逻辑、神经元的延时作用、利用神经元构造存储元件、模拟条件反射的神经网络.

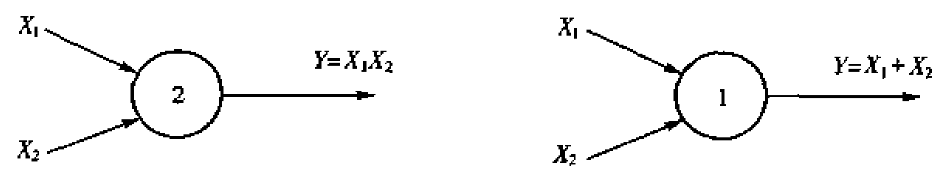


图 2.5　用 M-P 模型实现二元 Boole 逻辑

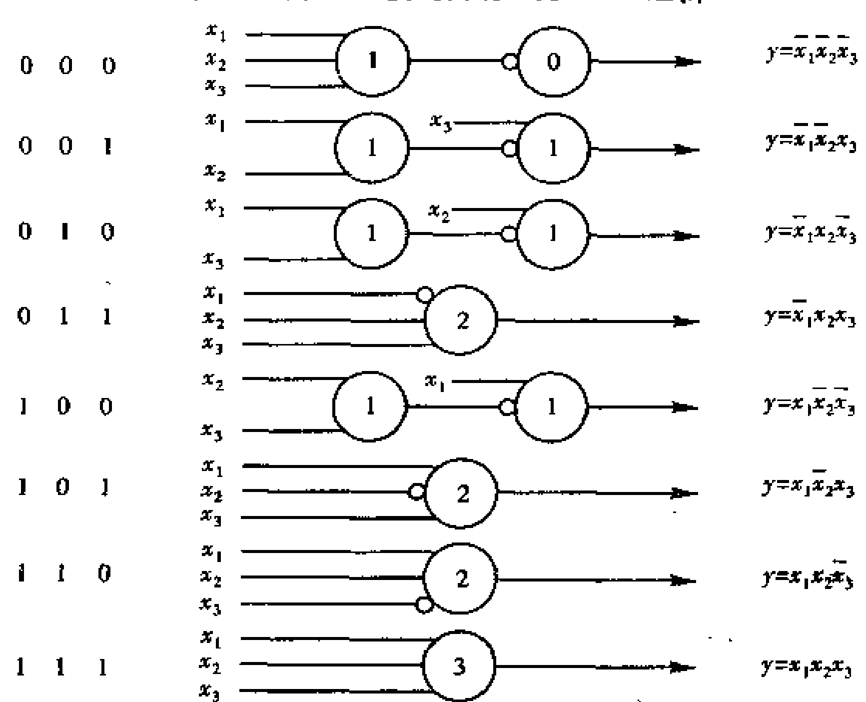


图 2.6　用 M-P 模型实现三元 Boole 小项逻辑

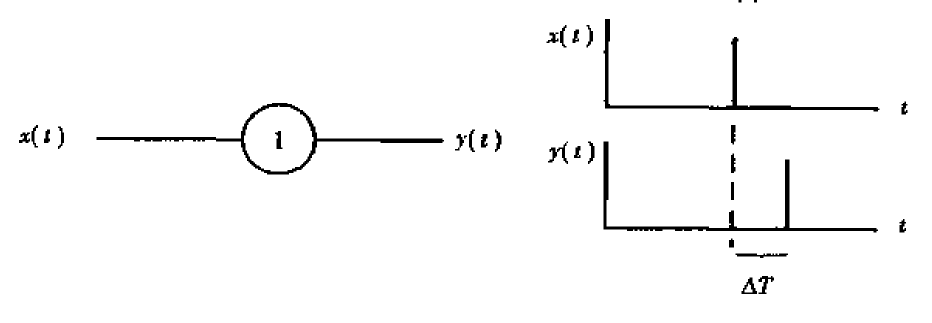


图 2.7　神经元的延时作用

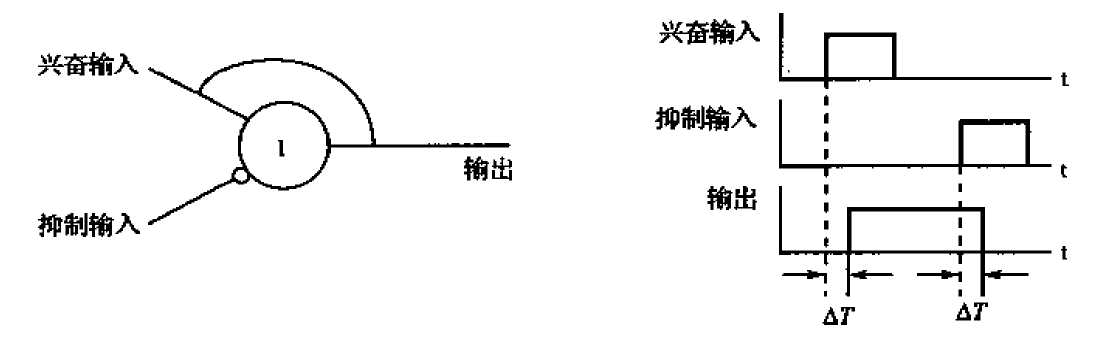


图 2.8 利用神经元构造存储元件

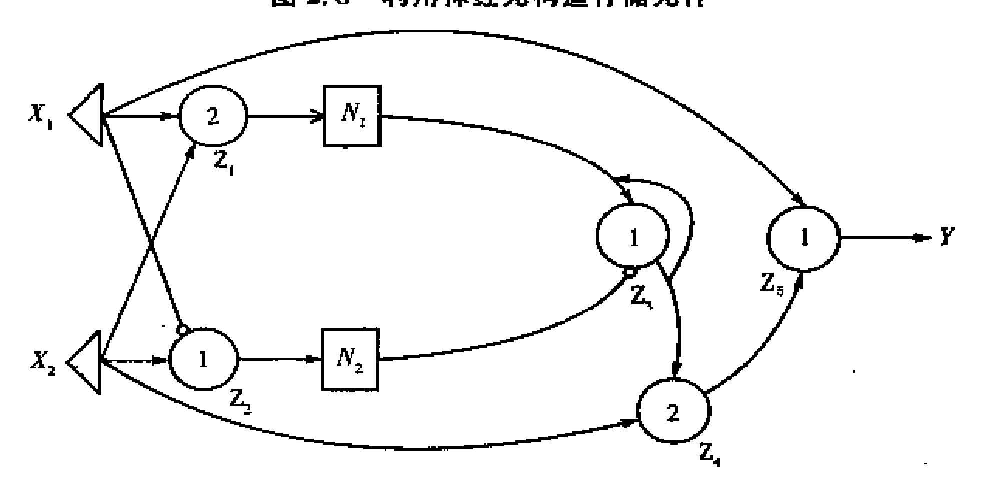


图 2.9 模拟条件反射的神经网络

考虑到信号通过一个神经元总是需要一定时间，可以把神经元看成是一个具有单位延时的神经组织.

这样利用神经元的延时作用和线性加权原理，就可以通过引入适当的反馈联接方式实现记忆功能.

利用神经元的适当模型可以构造具有某些复杂行为的网络系统，这就是神经网络. 早期的神经网络都比较简单，通常是用分立电子器件实现(即所谓硬件方法). 网络中只包含很少几个神经元. 模拟的行为也比较简单，不过都在某种程度上具有智能的特征.

图 2.9 是一个用来模拟条件反射的建立和消退的早期神经网络系统. 图中 X_1 代表无条件刺激（如巴甫洛夫试验中的"食物"），X_2 代表条件刺激或信号(如巴甫洛夫试验中的"铃声")，N_1 和 N_2 分别是限值为 N_1 和 N_2 的"达限归零"计数器，即当输入 N_1(N_2)个脉冲后，计数器产生一个脉冲输出，同时回零重新计数. 用数学语言来说就是一个 mod N_1(mod N_2)的模计数器. 图中其余的组件(共 5 个：$Z_1\cdots Z_5$)都是基于线性加权模型的神经元，它们的阈值和联接方式都已在图中注明.

对照图 2.9 可以看出，如果 X_1 和 X_2 同时输入 $N<N_1$ 次，那么，一方面由于 X_1 与 Z_5 有直接联系，Z_5 就会产生 N 次输出，表示 Y 在每次联合刺激下都产生兴奋；另一方面由于 X_1 与 Z_1 有联系，X_2 与 Z_5 有联系，因此 Z_1 也会产生 N 个脉冲送到模计数器 N_1，但因 $N<N_1$，故模计数器 N_1 不会产生输出；再者，由于 X_1 是 Z_2 的抑制性输入，于是 Z_1 不能兴奋，N_2 不计数，在这种情形下，Z_5 无输出，因此 Z_3 也不能兴奋. 这个过程相当于巴甫洛夫试验中建立条件反射的训练过程.

一旦当 $N=N_1$，则模计数器 N_1 产生输出脉冲，从而使 Z_3 兴奋(注意此时模计数器 N_2 仍然没有输入和输出)．一旦 Z_3 兴奋，由于存在反馈联接，将使 Z_3 维持兴奋状态(记忆)，Z_3 恒有输出送至 Z_4，在这一条件下，一方而由于 X_1 直接作用使 Z_3 兴奋，同时由于 X_2 与 X_3 的作用使 Z_4 产生输出也会使 Z_5 兴奋，这是条件反射建立起来的标志.

此后，如果只有 X_2 输入而没有 X_1 的输入，Z_5 也会产生兴奋($Y=1$)．这就是条件反射的效应.

但是，如果事件 $X_2\overline{X}_1$(即有铃声无食物)出现的次数达到 N_2 次，那么，由于 Z_2 的抑制性输入不起作用．Z_2 产生了 N_2 个脉冲送给模计数器 N_2，使模计数器 N_2 产生一个脉冲送给 Z_3，从而清除了 Z_3 的兴奋记忆，结果使 Z_4 不能兴奋，因此，Z_3 也不再能兴奋．这就是条件反射的消退.

§2.3 电子神经元

图 2.10 是一个典型的人工电子神经元．在这个人工电子神经元中，用一个电子放大器作为人工神经元的细胞体，这个作为细胞体的电子放大器是由一些二极管和三极管组合成的．放大器的输入端作为人工神经元的树突，可以用来输入信号，放大器的输出端可以看作人工神经元的轴突，它可以用来输出神经元的信号．在放大器的输入端和输出端联接了一些电阻器，这就好像是树突上的突触，这些电阻不仅可以用来传递信息，而且还能用来记忆信息，使电子神经元具有学习功能．因此，这些作为突触的电阻是可变电阻，也是设计人工电子神经元的关键．它们是代表输入点到神经元间的权重，并在外部控制信号作用下可改变其阻值．由此，图 2.10 的人工电子神经元也像真的生物神经元那样具有细胞体、树突、突触和轴突，而且也能像真的生物神经元那样，可以传递信息，还可以记忆信息，并具有学习本领．当然，这种人工神经元的学习功能，与生物神经元相比，那当然是差得很远了.

从人工神经元输入/输出的关系出发，一般可将人工神经元分成三种模式，图 2.11(a)是阶梯模式，当输入信号 X 大于等于某一个值 a 时神经元被激活，这时输出 Y 等于 1．其他时候，人工神经元处于非激活状态，输出 Y 等于 0．图 2.11(b)是准线性模式，即当输入信号大于某一个值 a 时，神经元被激活，这时输出 Y 等于 1．当输入信号 X 小于等于 0 时，神经元处于非激活状态，输出等于 0．当输入 X 介于 a 与 0 之间时，Y 与 X 是线性关系．图 2.11(c)是"S"曲线模式，人工神经元输入与输出之间满足 $Y=1/(1+e^{-X})$ 关系．由于"S"曲线模式表明人工神经元上一种非线性模式，目前受到科学家们的广泛重视.

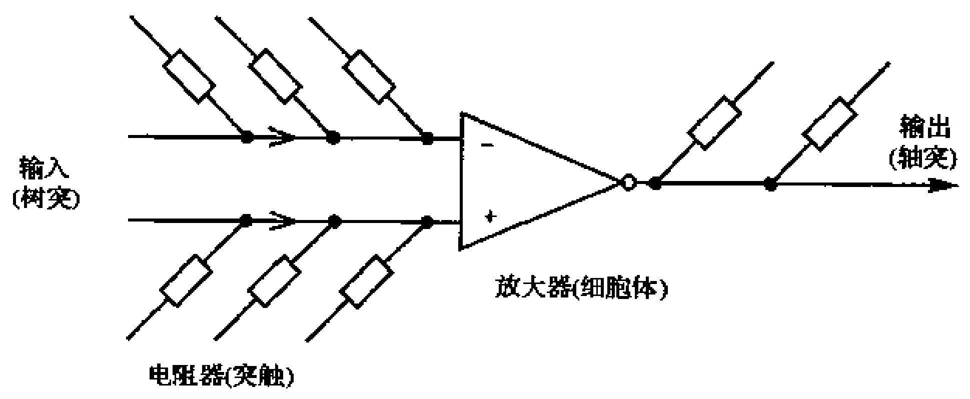


图 2.10 人工电子神经元

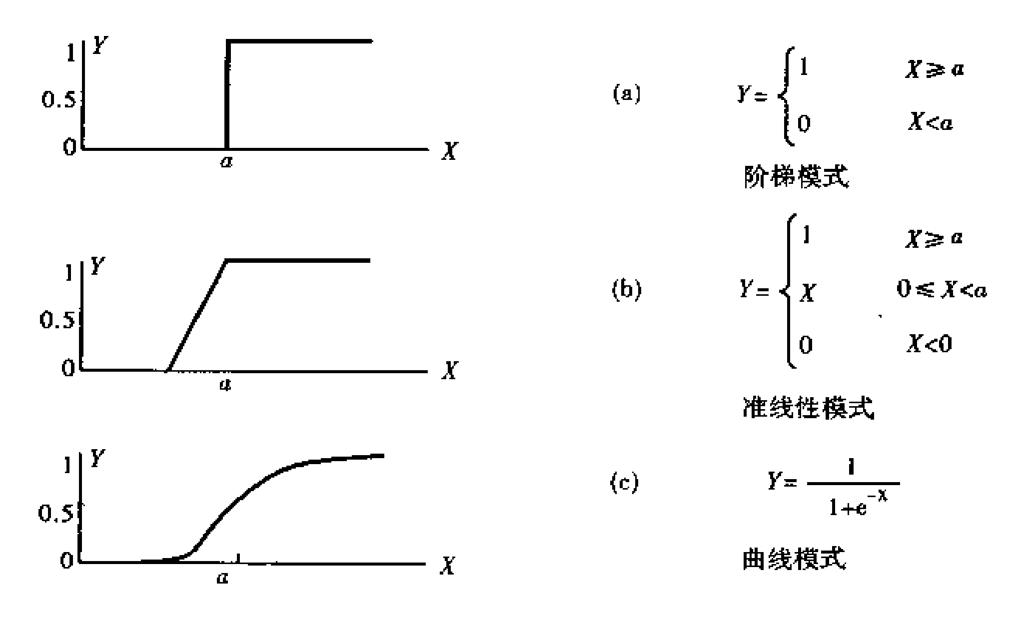


图 2.11　人工神经元网络的三种模式

§2.4　人工神经网络模型

人工神经网络定义：Kohonen T 指出人工神经网络就是由简单单元（通常为适应型）组成的广泛并行互连的网络，它的组织能够模拟生物神经系统的真实世界物体所做出的交互反映.

2.4.1　神经网络节点的形式化描述

如图 2.12 所示，设 u_i 为构成神经网络的某个神经元的内部状态，θ_i 为阈值，x_i 为输入信号，w_{ij} 表示从 u_i 到 u_j 连接的权值. s_i 表示外部输入信号（在某些情况下，它可以控制神经元 u_i，使它保持在某一状态），上述假设可描述为：

$$\sigma_i=\sum x_j w_{ij}+s_i-\theta_i \tag{2.4.1}$$

$$u_i=f(\sigma_i) \tag{2.4.2}$$

$$y_i=g(u_i)=h(\sigma_i),\quad h=gf \tag{2.4.3}$$

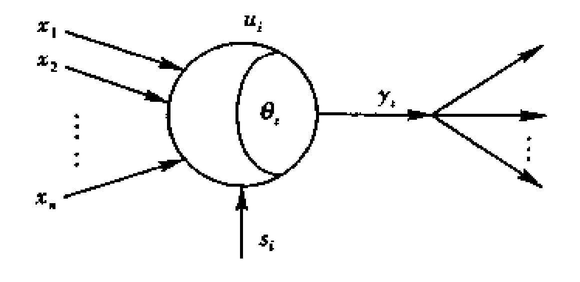


图 2.12　神经元结构模型

2.4.2 神经元状态转移函数的类型

当神经元没有内部状态时，可令 $y_i=u_i$，$h=f$. 如图 2.13 所示，常用的神经元状态转移函数有：

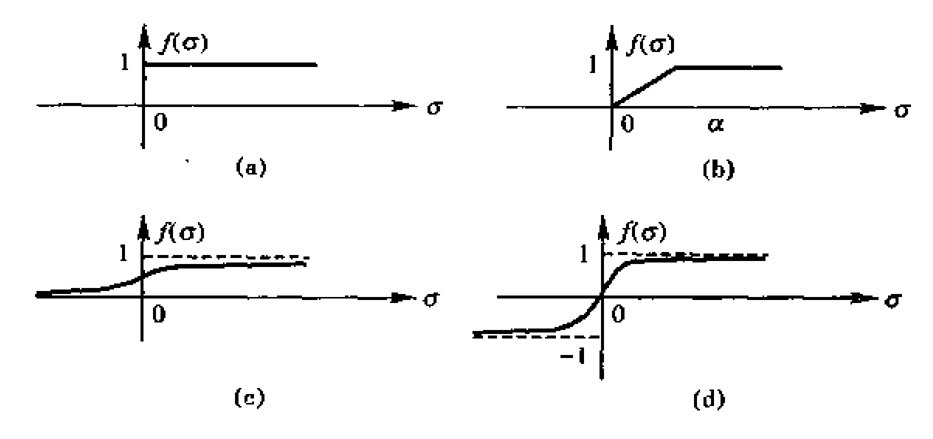


图 2.13 神经元状态转移函数

1. 阶跃函数

如图 2.13(a)所示，

$$y=f(\sigma)=\begin{cases}1 & \sigma\geqslant 0\\ 0 & \sigma<0\end{cases} \tag{2.4.4}$$

2. 准线形函数

如图 2.13(b)所示，

$$y=f(\sigma)=\begin{cases}1 & \sigma\geqslant\alpha\\ \sigma/\alpha & 0\leqslant\sigma<\alpha\\ 0 & \sigma<0\end{cases} \tag{2.4.5}$$

3. Sigmoid 函数

如图 2.13(c)所示，

$$f(\sigma)=\frac{1}{1+\mathrm{e}^{-\sigma}} \tag{2.4.6}$$

4. 双曲正切函数

如图 2.13(d)所示，

$$f(\sigma)=\mathrm{th}\sigma \tag{2.4.7}$$

2.4.3 神经网络分类及其拓扑结构

一、神经网络的分类

1. 按照对生物神经系统的不同组织层次和抽象层次的模拟分

(1)神经元层次模型：研究工作主要集中在单个神经元的动态特性和自适应特性，探索神经元对输入信息选择的响应和某些基本存贮功能的机理.

(2)组合式模型：它由数种相互补充、相互协作的神经元组成，用于完成某些特定的任务.如模式识别、机器人控制等.

(3)网络层次模型：它是由许多相同神经元相互连接而成的网络，从整体上研究网络的集

体特性.

(4)神经系统模型:一般由多个不同性质的神经网络构成,以模拟生物神经的更复杂或更抽象的性质.

(5)智能型模型:这是最抽象的层次,多以语言形式模拟人脑信息处理的运行、过程、算法和策略.这些模型试图模拟如感知、思维、问题求解等基本过程且与AI相关.

2.按照网络的结构与学习方式分

(1)按照网络的性能分:连续型与离散型网络,确定型与随机型网络.

(2)按照网络的结构分:前馈网络,反馈网络.

(3)按照学习方式分:有教师指导的网络,无教师指导的网络.

(4)按照连接突触的性质分:一阶线性关联网络,高阶线性关联网络.

二、神经网络的拓扑结构

1.不含反馈的前向网络

如图2.14(a)所示,神经元分层排列,组成输入层、隐层(亦称中间层,可有若干层)和输出层.每一层的神经元只接受前一层神经元的输入.输入模式经过各层的顺次变换后,得到输出层输出,感知器和误差反向传播算法中所使用的网络都属于这种类型.

2.从输出层到输入层有反馈的前向网络

如图2.14(b)所示,它可用来存贮某种模式序列,如神经认知机.

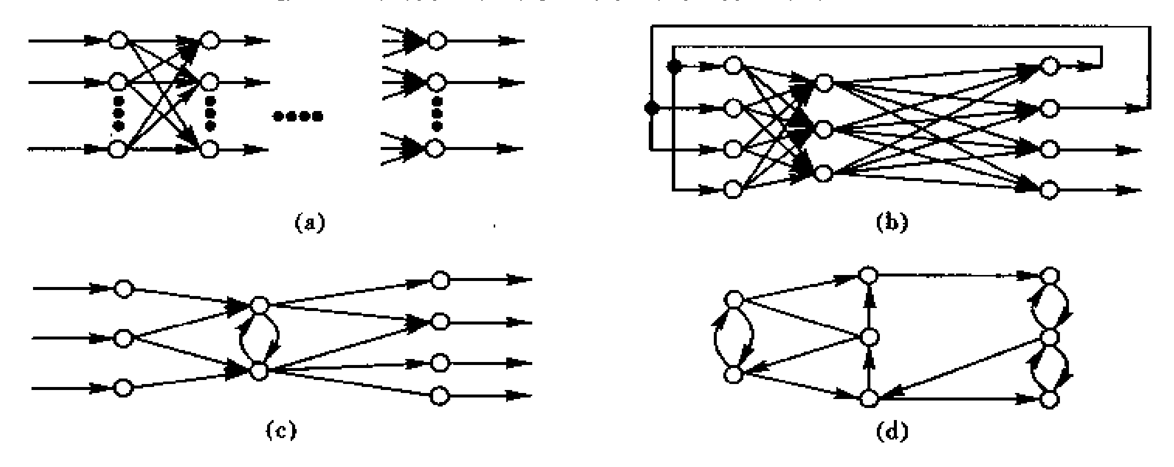


图2.14 网络结构的各种形态

3.层内有反馈的前向网络

如图2.14(c)所示,通过层内神经元间的相互结合,可以实现同一层内神经元之间的横向抑制或兴奋机制,这样可以限制每层内能同时动作的神经元数,或者把每层内的神经元分成若干组,让每组作为一个整体来动作.例如,可以利用横向抑制机制把某层内具有最大输出的神经元挑选出来,而抑制其他神经元处于无输出状态.

4.相互结合型网络

如图2.14(d)所示.这种网络是在任意两个神经元之间都有可能有连接.HNN和Boltzman机都属于这一类.在无反馈的前向网络中,信号一旦通过某个神经元,过程就结束了.而在相互结合网络中,信号要在神经元之间反复往返传递,网络处在一种不断改变状态的动态之中.从某个初态开始,经过若干次的变化,才会达到某种平衡状态.根据网络的结构和神经元的特性,还有可能进人周期振荡或其他如浑沌等平衡状态.

以上 1,2 和 3 看作是 4 的一种特殊情况,但不论从网络的计算和学习机制看,还是从它们应用的场合看,都有很大的区别.

表 2.1 反馈 ANN:无教师指导学习

范式	在线	离线	模式匹配	学习方式	限制	典型应用
ART1	是	否	是	Hebbian	限于二进制类型	复杂系统分类
DHNN	不是	是	否	Hebbian	限于二进制类型	语言处理
CHNN	不是	是	是	Hebbian 或竞争式	有限存贮	组合优化
BAN	不是	是	是	Hebbian	离线计算机	图像识别
空域 AM	不是	是	不是	Hebbian	离线模拟	存贮域模式
ABAM	是	不是	是	Hebbian	有限存贮	连续时间模式处理
CABAM	是	是	是	Hebbian	有限存贮模拟	组合优化
FOM	是	是	不是	Hebbian	离线模拟	组合优化

表 2.2 无反馈 ANN:有教师指导学习

范式	在线	离线	模式匹配	学习方式	限制	典型应用
Adaline	否	是	是	误差纠错	离线处理	预测噪声
Madaline	否	是	是	误差纠错	离线学习	噪声消除
BP	否	是	是	误差纠错	离线学习	特征识别
CPN	否	是	是	Hebbian	缺乏控制	自规划
BM/CM	否	是	是	Hebbian 与模拟退火	离线学习	组合优化

表 2.3 前向 ANN:有教师指导学习

范式	在线	离线	模式匹配	学习方式	限制	典型应用
LM	否	是	是	Hebbian	线性处理	过程控制
LAM	否	是	是	Hebbian	有限存贮	系统控制
最优线性 AM	否	是	是	Hebbian	离线处理	信号处理
FAM	是	不是	是	Hebbian	有限存贮	知识处理
盒中脑 BBB	不是	是	是	纠错	离线模拟	实时分类
感知器	否	是	是	纠错	线性可分	预测

2.4.4 神经网络的知识表示与处理能力

一、神经网络的知识表示

1. 知识(概念、信息)的存储形式

20 世纪 80 年代,费德曼(Feldman)等人提出在人工神经网络中采用连接制网络模型(Connectionist Model). 它不是把知识(或概念)保存在存贮器中,而是由组成人工神经网络计算机的各个神经单元之间的一些连接和连接强度来保存. 这里的连接强度,在连接制模型中称为"权重". 例如在人工神经网络中要保存"他是工人"这样一个概念,就需要有一个神经元代表"他",一个神经元代表"工人",另有一个神经元代表"是". 当这三个神经元之间的连接强度都同时活动时,便代表这个概念的输出. 如果要增加新的"知识",人工神经网络就必须增加新的神经元及它与各有关神经元之间的连接. 数以万计的神经元互相连接,便构成了人工神经网

络.

因此，连接制网络模型的特点是：网络中的每个神经元只记少量的信息. 它们只对输入的数据进行简单的逻辑运算或算术运算，并把结果送到有关的连线上. 这些操作不由外部程序决定，可以由这些神经元自主地进行.

2. 知识表示和神经元数量间的关系

神经元之间用一个逻辑值（传递的标志）作为信息交换的网络称为递号（Mark-passing）网络，若用一个标量值作为信息交换的网络称为递值（Value-passing）网络.

这样，在人工神经网络中，“知识”不是由一个神经元来代表，而是由一组神经元来代表，即前面所述的分布式表示法. 如果用很少的神经元来表一个“知识”，会出现很高的相同编码概率，即造成“知识”的混淆. 韦肖（Willshow）证明，表示一个“知识”的神经元数量，最好是网络中神经元总数的指数值，即：

$$Y = 2^X$$

式中： Y——网络中神经元的总数；

X——一个“知识”的神经元数.

若网络中总共有 10^6 个神经元，则每个“知识”最好用 20 个神经元来编码.

在连接制网络中，神经元之间的连接是动态的. 由于每个神经元可以同时输入信号进行处理，所以整个网络可以同时获取大量的知识. 人工神经网络计算机可以根据这些“知识”的连接权重来选择最合适的答案.

二、神经网络的工作方式

1. 学习期

此时各计算单元状态不变，各连接权值可修改（通过学习样本或其他方法）.

2. 工作期

此时各连接权值固定，计算单元的状态变化，以求达到稳定状态.

3. 工作方式

（1）利用能量函数的所有极小点：能量函数的所有局部根小点都起作用. 这一类主要用于各种联想存储器、信息压缩及编码，通常称为 HNN 的属于这一类，也包含一些自组织网络.

（2）利用能量函数的全局最小点：只利用能量函数的全局最小点，这一类主要用于求解组合最优化问题. 例如 TSP 问题以及视觉、控制等领域中的一些反演问题.

（3）利用映射函数：在工作中不考虑能量函数，主要作用是函数映射，它主要包括用于模式分类和特征抽取的多层感知器.

三、神经网络的信息处理能力

1. 信息的分布式存储与分布式处理

在人工神经网络系统中，信息的存贮与处理（计算）是合二为一的，即信息的存贮体现在神经元互连的分布上，并以大规模并行分布式方式处理. 这种并行处理决不是简单地以“空间复杂性代替时间复杂性”，而是反映不完全相同的“计算”原理. 从数学观点看，可以把神经网看作是由大量子系统组成的大系统，系统的最终行为完全由它的吸引子决定，如果视动力系统的稳定吸引子为记忆的话，那么从初态向吸引于流动的过程就是寻找记忆的过程. 初态可以认为是给定有关记忆的部分信息. 换言之，流动的过程就是从部分信息中找出全部信息的过程，这就是联想记忆的基本原理. 进一步，若视动力系统的稳定吸引子为系统计算能量函数的最小点，

系统最终会流向期望的最小点,“计算”也就在运动过程中悄悄地完成了.运动的时间就是计算时间,这就是神经网络计算机的基本原理.

2.神经网络处理信息的特点

神经网络系统与现代数字计算机有如下不同:

(1)以大规模模拟并行处理为主,而现代数字计算机只是串行离散符号处理.

(2)具有很强的鲁棒性和容错性,善于联想、概括、类比和推广,任何局部的损伤不会影响整体结果.

(3)具有很强的自学能力.系统可在学习过程中不断地完善自己,具有创新特点,这不同于AI中的专家系统,后者只是专家经验的知识库,并不能创新和发展.

(4)它是一个大规模自适应非线性动力学系统,具有集体运算的能力,这与本质上是线性系统的现代数字计算机迥然不同.

3.神经网络处理信息的应用领域

从广义角度讲,微积分、文字翻译、推理等都是一计算过程,而从数学观点看,计算就是在满足一定的自然规则,在某种硬件上所发生的一些物理规则.因此,计算可表示为一动力系统中的状态空间变换的轨迹.神经网络的计算就是其中状态的转换,其计算过程可以认为是状态的转换过程,对给定的输入,其计算结果即是系统的稳定状态.

迄今为止开发的30余种神经网络模型都是针对某种特殊用途的,因而对这些特殊问题是有很强的计算能力.现在,人们还在努力寻找新的机制以构造通用神经网络模型.目前,神经网络至少可以完成以下信息处理任务:

(1)数学逼近映射:开发合适的函数 $f:A\subset\mathbf{R}^n\rightarrow B\subset\mathbf{R}^m$,以自组织的方式响应以下的样本集合:$(x_1,y_1)\cdots(x_m,y_m)$,(这里 $y_i=f(x_i)$ 或 $y=f(x)+N$,其中 N 为噪音过程).这里描述的当然是一般的数学抽象.像识别与分类这些计算都可以抽象为这样的一种近似数学映射.BPN和CPN模型都可以完成这种计算.

(2)概率密度函数的估计:通过自组织的方式开发出一组等概率“锚点”来响应在空间 $\mathbf{R}^n$ 中按照一定确定概率密度数 p 选取一组向量样本 $x_1,x_2,x_3,\cdots$.CPN和SOM模型可以完成这种计算.

(3)从二进制数据基中提取相关的知识:将从二进制基中提取的相关知识形成一种知识的聚集模型.这些知识依照数据基的自组织在它们之间有某种统计上的共性,并依这些共性来响应输入的数据基.BSB有能力进行这种计算.

(4)形成拓扑连续及统计意义的同构映射:这是对固定概率密度函数选择输入数据进行自适应的一种自组织映射.终归使得数据空间上的不同项有某种同构.SOM模型最先计算这类问题.

(5)最近相邻模式分类:通过比较大量的存贮数据来进行模式分类,但首先对样本模式进行分类.这种能力可应用于暂态或暂稳态模式分类,并且可用层次性的存贮模式来表达存贮.绝大多数的神经网络模型均能进行这种计算,比如:ART、AVABAM、BCM、BPN、BSB、CBD、CPM、HCP、LRN、MDL、HEO、PTR.

(6)数据聚类:这是采用自组织的方式所选择的“颗粒”或模式的聚类,以此来响应输入数据.聚类是可变的,但要限制其鞍点的个数.对于任何新的目标,只要在系统中没有给其提供聚类,都要形成新的聚类.很显然,这种能力可直接应用于雷达的多目标跟踪,ART模型最适于

这种计算.

(7)最优化问题的计算:这是用来求解局部甚至是全局最优解的,HOP 模型、BCM 模型有能力进行这种计算.

§2.5 人工神经网络的学习规则

如何调整网络的权系,使网络能得到希望的映射函数或使网络从初态向稳态转换.

2.5.1 人工神经网络的学习方式

(1)死记学习:联想记忆(自、他)是先设计成记忆模式为稳态,给定有关信息时就回忆起来.如 HNN 网络.

(2)有教师指导的学习:给定输入模式,教师指定期望输出,通过调整权系,使实际输出与期望输出达到一致.如前向网络的 BP 算法.

(3)竞争学习:给定一个输入模式,使某些(个)神经元兴奋.如 ART 网络.

2.5.2 人工神经网络的学习规则

学习规则可分为以下几类:

一、相关规则

依据连接之间的激活水平改变权系(死记学习).

1. Hebb D D 学习规则

网络中若第 i 与第 j 个神经元同时处于兴奋状态,则它们之间的连接权值 W_{ij} 应当加强,即:

$$W_{ij}(n+1) = W_{ij}(n) + \Delta W_{ij} \tag{2.5.1}$$

$$\Delta W_{ij} = \alpha Y_i Y_j, \qquad \alpha > 0$$

其中 α 为学习速率系数,要求:

$$Y \in \{0,1\} \quad 或 \quad Y \in \{-1,+1\}$$

Hopfield 网络使用修正的 Hebb 规则:

$$\Delta W_{ij} = (2Y_i - 1)(2Y_j - 1) \tag{2.5.2}$$

要求:

$$Y \in \{0,1\} \quad 或 \quad Y \in \{-1,+1\}$$

二、纠错规则

依据输出节点的外部反馈来改变权系,使实际输出与期望输出相一致(有教师指导的学习).

2. 感知器的学习规则

(1)如果一个节点的输出正确,则连接权值不变.

(2)如果输出本应为 0 而为 1,则相应地减小权值.

(3)如果输出本应为 1 而为 0,则相应地增加权值.

3. 学习规则

网络中神经元 j 与神经元 i 的连接权值为 W_{ij},则对权值的修正为:

$$\Delta W_{ij} = \eta \delta_i Y_j \tag{2.5.3}$$

其中：η——学习率；$\delta_i = T_i - Y_i$ 为 i 的偏差，即 i 的实际输出和教师信号之差；

δ——学习规则，仅用于单层网络的学习规则，如单层感知器的学习.

4. 广义 δ 学习规则

用于多层网络，对于输出层节点和相邻隐层节点间的权值修正用 δ 学习规则，对于其他层间的连接权值，则使用广义 δ 学习规则. 设 i 为隐层节点，其偏差的计算为：

$$\delta_i = f(\text{net}_i) \sum \delta_k W_{ki} \tag{2.5.4}$$

$$\Delta W_{ij} = \eta \delta_i Y_j \tag{2.5.5}$$

其中：δ_k——i 的上一层节点的偏差；

W_{ki}——i 与 k 间的连接权值. i 的下一层节点的偏差可以用递归的方法得到.

5. Boltzmann 机学习规则（模拟退火算法）

模拟递火算法是应用于神经网络中的一种重要的算法，该算法能找到全局最优解. 1953 年 Meropols N 等人研究二维相变时提出模拟退火算法，1983 年 Hinton G E，Sejnowski T J 和 Ackley D H 把这种方法用于神经网络，提出了 Boltzmann 机. 1984 年 Geman S 和Geman D 给出了退火率（温度随时间变化率）为：$T(t)/T(0) \propto 1/\ln t$，其中 $T(0)$ 是初始温度. 这个退火过程太慢，因而效率低，难于实用. 1985 年 Harold H Szu 提出了一种快速模拟退火算法，称为 Cauchy Machine，使这种方法有了实际应用的可能性，退火率为：$T(t)/T(0) \propto 1/t$.

模拟退火算法基本上由三部分组成：

(1) 以一定的概率密度函数跃迁到新的状态，这个概率密度函数称为生成函数.

(2) 以一定的概率密度函数容忍评估函数的偶然上升，这个概率密度函数称为容忍函数.

(3) 以一定的冷却方式降低温度，这个等效温度是生成函数和容忍函数的控制参量.

经典的模拟退火算法中使用了高斯型生成函数，

$$G_g(x) \approx \exp\left(-\frac{x^2}{T(t)}\right) \tag{2.5.6}$$

其中 $T(t)$ 是温度，决定了概率密度分布的特征宽度，这种分布函数远处是指数型衰减的，因而代表一种局域型搜索过程，为了能找到全局最小，温度要下降得很慢，Geman 兄弟证明只要满足退火率

$$\frac{T(t)}{T(0)} = \frac{1}{\ln t} \tag{2.5.7}$$

且 $T(0)$ 要足够大.

算法大致如下，从一个随机选取的状态出发，依据生成函数产生一个新的状态，如果这个新的状态的评估函数值比原来的状态低，则令它为系统新的状态；如果它比原先状态的评估函数值高，则它成为新状态的概率由容忍函数确定，若系统不进入这个状态，则它仍保持原先的状态，生成函数与容忍函数按照规定的冷却方式变化.

在学习过程中，随机地改变一权值，确定权值改变后产生的最终能量，并按照下述规则来确定是否保留此权值的变化.

(1) 若随机改变权值后，能量降低，则保留这一改变.

(2) 若随机改变权值后，能量没降低，则根据一预选的概率分布来保留这一改变，否则

(3) 拒绝权值的改变，使权值恢复到原来的值.

在第(2)步虽然性能有可能变差，但从整体上来看，却能保证能达到能量最小点. 在(2)中逐渐减小概率值.

6. 梯度下降算法

将数学上的最优化方法应用于 ANN 中，权值的修正量正比于误差对加权的一阶导数.

$$\Delta W_{ij} = -\eta \frac{\partial E}{\partial W_{ij}} \tag{2.5.8}$$

其中：E——描述误差的误差函数；

η——学习率.

三、无教师指导的学习规则

学习表现为适应于输入空间的检测规则.

7. 竞争学习规则

在学习过程中，神经元均参与彼此间的竞争活动，具有最大输出的节点是获胜者. 该获胜节点具有抑制其竞争者的能力和激活邻近节点的能力. 只有获胜节点和其邻近节点的权值允许被调整. 获胜节点的邻近区域(胜域)在学习过程中逐渐变小.

设输入层有 n 个节点，输出层 m 个节点，d_j 为距离接近测度，则：

$$d_j = \sum_{i=1}^{n}(X_i - W_{ij})^2 \quad i \in \{1,2,\cdots,n\}, j \in \{1,2,\cdots,m\} \tag{2.5.9}$$

胜者

$$d_j^* = \min(d_j), \qquad j \in \{1,2,\cdots,m\} \tag{2.5.10}$$

$$\Delta W_{ij} = \begin{cases} \alpha(X_i - W_{ij}), & j \in N_c \\ 0, & j \notin N_c \end{cases} \tag{2.5.11}$$

§ 2.6 人工神经网络与传统计算机的比较

2.6.1 人工神经网络计算机和冯·诺依曼计算机的比较

对于人工神经网络计算机与冯·诺依曼计算机的主要不同可见表 2.4.

表 2.4 人工神经网络计算机和冯·诺依曼计算机的比较

性能比较	冯·诺依曼计算机	人工神经网络计算机
运行方式	串行、线性系统	并行、非线性动力学系统
存储方式	集中存储、存储器中	分布存储、连接权值中
信息处理方式	编程、符号计算、逻辑推理、逻辑思维、符号主义	非编程、联想记忆、类比、形象思维、连接主义
知识获取	编程实现	用例子对系统进行训练
学习能力	不能创新和发展	有自学习能力
问题求解	算法公式	结构的选择和一组代表性例子的定义
可靠性	差	鲁棒性强、容错性好
计算精度	高精度	低精度、非线性映射
运算速度	快	慢

2.6.2 人工神经网络和人工智能的比较

一、人工神经网络(ANN)和人工智能(AI)的比较

(1)人工智能系统是在传统计算机硬件基础上发展起来的图灵过程软件,以使传统的计算机更有用和更有智慧.

(2)人工神经网络是把算法和结构统一为一体的系统,这是一种硬件和软件的混合体,由于它在某种程度上模拟大脑的结构,所以这种系统有更高的智慧.

(3)两者在运算逻辑上实际上是相同的,例如,识别事物均采用等价类模型,以识别苹果为例,当水果、红色及美味三要素重叠时,便判定为苹果.

二、人工神经网络(ANN)和人工智能(AI)在工作原理上的不同

(1)知识表象不同. AI 用一维串表示知识,ANN 则用二维或高维矩阵表示知识,优点如下:

①由于自由度增加,使存储空间扩大,因而存储容量可以更大.

②容错性大为提高,因为高维空间中每一状态有更多的近邻,使多体效应更加复杂和显著. 另外,多维空间会有相变,对知识的存储和学习有重要意义.

③高维空间更易于分类,分类就是识别事物,其原则是使得同类事物更加聚集,异类事物更分离,此原则在高维空间中容易实现.

(2)学习方式不同. AI 要求预编程,它是在指定规则的基础上工作的,这种系统只能进行推理. ANN 是通过学习建立和改变知识,它具有推广和抽象的能力,这种系统可以进行引证,无需预编程和制定工作规则.

2.6.3 人工神经网络与生物系统的区别

由于 ANN 是以模拟生物神经系统为基础的,因此比较 ANN 和生物神经系统的区别对于未来的研究是有实际意义的.

(1)数字和逻辑电路的原理不适用于生物神经系统,因为生物神经系统既没有统一的时钟来同步各种神经元的翻转,也没有硬性限制一次只能有一个神经元改变状态的异步逻辑. 由此看来,模拟电路与实际神经元比较接近.

(2)生物神经元及突触都不是双稳态记忆元件.

(3)生物神经系统中没有机器指令,也没有时序控制码.

(4)生物神经系统不是自动机,不能完成递归运算.

(5)生物神经系统不拥有专门问题求解、抽象化或决策电路.

(6)即使在最高层次上,生物神经系统的信息处理方式也与数字电路不同.

§2.7 人工神经网络的发展方向与研究问题

ANN 的研究方兴未艾,要准确地预测其发展方向是非常困难的. 目前,ANN 模型首先必须解决可编程性、全局稳定性、结构稳定性等问题. 今后的研究工作至少包含以下一些基本内容.

2.7.1 人工神经网络模型的研究

人脑的生理结构，即神经网络原型的研究.

人脑思维的机制，特别要从信息科学和认知科学的角度来阐明这种机制.

神经元生物特性如时空特性、不应期、电化学性质等完善的人工模拟，如高阶非线性模型、多维局域连接模型.

神经网络计算模型，特别是统一化的便于实现的模型.

神经网络学习算法与学习系统.

2.7.2 人工神经网络基本理论的研究

非线性内在机制——自适应、自组织、协同作用、突变、奇怪吸引子与浑沌、分维、耗散结构、随机非线性动力学等.

神经网络的基本特性——稳定性、收敛性、容错性、鲁棒性、动力学复杂性.

神经网络的能力与判别准则——计算能力、准确性、存贮容量、准则的表达与综合性能判别.

面向应用的网络设计与综合——专用和通用神经网络计算机的设计、单元连接、运算模式、I/O、存贮/计算，与现有技术的兼容与匹配等.

2.7.3 人工神经网络智能信息处理系统的应用研究

自适应信号处理——自适应滤波、时间序列预测、均衡、谱估计、阵列处理、检测噪声相消等；

非线性信号处理——非线性滤波、非线性预测、非线性谱估计、非线性编码、映射、调制、解调、中值预处理等；

优化与控制——优化求解、辨识、鲁棒性控制、自适应、变结构控制、决策与管理、并行控制、分布控制、智能控制等；

认知与 AI——模式识别、计算机视觉、听觉、特征提取、语言翻译、AM、逻辑推理、知识工程、专家系统、智能计算机与智能机器人、故障诊断、自然语言处理等.

2.7.4 人工神经网络计算机

一、目前神经网络计算机的实现

通用计算机的软件仿真模拟；在通用计算机上插入专用的有电子神经元电路的接口板，再配以软件支持；专用的神经网络并行计算机. 20 世纪 90 年美国的人工智能实验室的希尔斯等人研制成功 CONNECTION-2 型人工神经网络计算机已投人使用.

尽管人们对于神经网络计算机的研究已取得相当成果，但由于人类对子人脑的认识还相当粗浅，对于建立一套完整的理论体系还是远远不够的，建立完全由神经网络构成的具有较高智能的计算机，目前来说还为时过早.

二、神经网络计算机的应用现状

目前，神经网络计算机已广泛地应用在社会生活和生产实际的各个领域. 如自适应信号处理；时间序列预测；非线性信号处理；组合优化与智能控制；决策与管理；模式识别；计算机视

觉、听觉；语言翻译；机器学习；专家系统；智能机器人；故障诊断；自然语言理解等. 例如，计算机味觉，计算机听觉.

关于智能本质的研究是自然科学和哲学的重大课题之一. 对于智能的模拟和机器的再现，可能会开拓发展出一代新兴产业. 由于智能本质的复杂性，现代智能的研究已超越传统的学科界限. 成为脑科学、神经学、心理学、认知科学、信息科学、微电子学，乃至数理科学共同关注的焦点. 人工神经网络的重大进展有可能使包括信息科学在内的其他学科产生重大的突破. 展望人工神经网络的成功应用，人类智能有可能产生一次新的飞跃.

三、光学神经网络计算机

21 世纪，随着人们探索自然界和人类社会中一系列非线性行为的规律，对于高水平信息处理计算机的需求日益增加. 而光学神经网络计算机作为能够模拟人脑功能的超分散、超并行的新一代计算机正从希望走向现实. 1991 年美国贝尔实验室公布了数字光学计算机的成果，据预测，21 世纪初光学计算机将问世.

1. 光学计算机的特点

光学计算机的运算速度会比今天的超级计算机快一千到一万倍；光在长距离传输要比电子数字信号快约一百倍；光学元件功耗低；无电磁辐射，可靠性强，保密性好.

2. 光学计算机的研究现状

目前，光学逻辑器件，包括非线性光逻辑器件和编码光逻辑器件、半导体双稳态、光于存储器、光开关、面发光激光器、光/电寻址空间光调制器等主要器件均已相继问世.

(1)二维图像联想存储的光学神经网络计算机. 1987 年，由美国加州工学院的 Psaltis D 等人研制成功，主要用于图像的快速光模式识别.

(2)二维可编程光学神经网络计算装置. 由美国宾州的俞教授领导的小组研制成功，采用高分辨率的视频监视作为联想存储器，用于不完整字母的识别.

(3)混合式人工神经网络计算系统. 1990 年，由美国卡内基梅隆大学的 Casasent D 等人提出. 该系统是用数字神经网络实行训练和学习，用光学神经网络进行快速在线处理.

四、分子生物计算机

分子生物计算机是一种利用蛋白质和其他各种大分于所组成的模拟人脑神经网络系统的全新计算机. 1983 年 11 月，美国国家科学基金会邀请了各国的 40 多位专家研讨有关生物分子计算机的理论与实验技术等问题. 之后，美国、日本等国都投人了大量的人力、物力和财力实施研究. 例如，日本从 1984 年起，每年投人 80 亿日元对分于开关、分子逻辑进行开发研究. 1990 年，东芝公司研制成功一种名叫人工视网膜的分子传感器，它有很强的图像信息处理能力. 虽然分子生物计算机的问世还需要一段相当长的时间，但是它的最终实现却是可能的.

1. 分子生物计算机的特点

(1)分子计算由蛋白质和它们所综合的系统结构实现，即不需要像数字计算机那样编程；

(2)分子活动的大规模并行特性可以提高计算效率，故比电子开关慢五个量级的单分于速度也不会影响整体的运算效率；

(3)分子的计算形式属于一种基质感触识别，因此它依照几何方法可以实现与上下文有关的自适应信号输人，而无需一位一位的串行处理.

2. 分子生物计算机的实现初探

分子生物计算机是受当代计算机的启发，从分子电子线路设计与造型出发，形成一个体积

小、速度快、存储量大的数字分子计算机.包括分子开关、分子逻辑和分子局部装配等的研究.

分子生物计算机是模仿生物的图像信息处理功能,即快速的识别、自组织和自学习能力的感触式分子计算机.包括感触识别的基本模式、模拟感触处理功能的几种支撑技术以及同现代计算机相互结合的研究.例如,将生物分子处理器作为数字计算机的预处理感觉装置,即将每个生物分子处理器做成不同分辨率的探测器阵列,应用分子感触式处理方法快速完成输入信号的特征辨识,然后再进行更高层次的运算,这种结构称之为生物分子预处理结构.再如,分子自适应存储结构,利用神经元间的连接和切换原理可以模仿大脑的自适应记忆存储/检索功能.

第三章　前馈型神经网络

许多科学家通过大量的观察和研究，认为人脑的认识过程不仅是按高度并行的网络结构进行，而且其神经系统还是一个分层结构. 由低层网络把处理后的信号提供给高层次的网络汇总选择. 经简化、抽象后的阶层网络结构参见图 2.14(a). 前馈型(feedforward type)网络是由输人层，若干中间层(隐蔽层)和输出层构成，层内各单元间无连接，层间的连接是由输人层经中间层至输出层. 信号一个方向流动. 前馈型网络为无反馈有教师信号的网络，又称为阶层网络(Layered network). 本章将介绍几种典型的前馈型网络的结构特点、功能及算法.

§3.1　感知器

感知器(Perception)是由美国学者 Rosenblatt F 于 1957 年提出的一个具有单层计算单元的神经网络. 它在识别印刷体字符方面表现出了良好的性能，引起人们很大的兴趣. 后来许多改进型的感知器在文字识别、语音识别等应用领域取得了进展，使得早期的神经网络的研究达到了高潮.

感知器的输人可以是非离散量，它的权向量不仅是非离散量，而且可以学习调整. 感知器是一个线性阈值单元组成的网络，可以对输人样本进行分类，而且多层感知器，在某些样本点上对函数进行逼近，虽然在分类和逼近的精度上都不及非线性单元组成的网络，但是可以对其他网络的分析提供依据.

3.1.1　单层感知器

一、单层感知器网络

图 3.1 为一单层感知器神经网络，输人向量为 $\boldsymbol{X}=(X_1,X_2,\cdots,X_n)$，输出向量为 $\boldsymbol{Y}=(Y_1,Y_2,\cdots,Y_m)$. 最简单的感知器仅有一个神经元，见图 3.2.

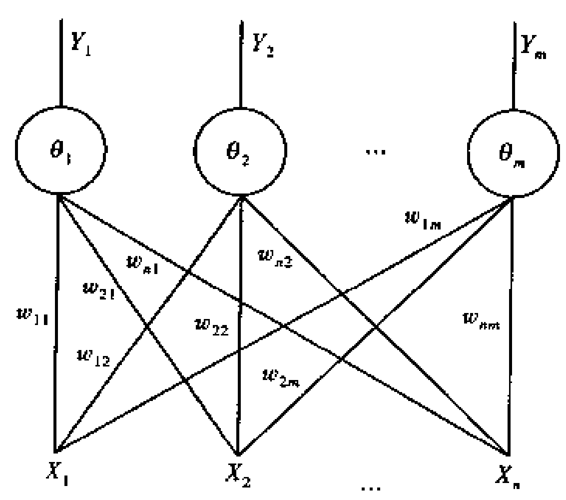


图 3.1　单层感知器网络

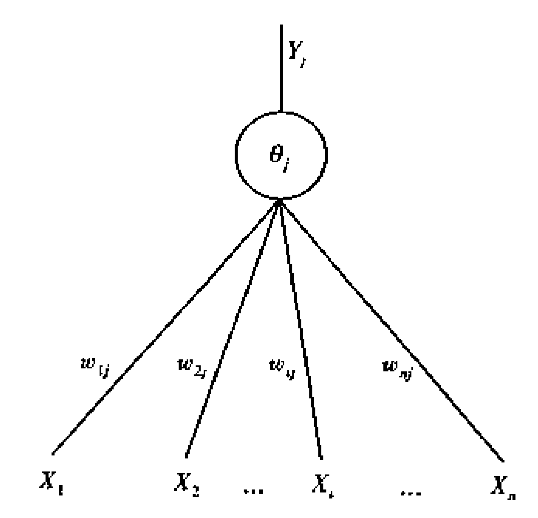


图 3.2　最简单的感知器

单结点感知器的输入向量为 $X\in \mathbf{R}^n$，权值向量为 $W\in \mathbf{R}^n$，单元的输出为 $Y\in\{-1,1\}$. 其中：

$$Y = f\left(\sum_{i=1}^{n} X_i W_i - \theta\right) = f(XW^{\mathrm{T}} - \theta) = f(X'W'^{\mathrm{T}}) \tag{3.1.1}$$

其中，$X'=(X,-1)$，$W'=(W,\theta)$.

$$y = \operatorname{sgn}(X'W') = \begin{cases} 1 & X'W'^{\mathrm{T}} \geqslant 0 \\ -1 & X'W'^{\mathrm{T}} < 0 \end{cases} \tag{3.1.2}$$

与 M-P 模型不同的是，权值 W 可以通过学习训练而加以改变.

二、单层感知器的学习算法

令 $W_{n+1}=\theta$，$X_{n+1}=-1$，则

$$Y = f\left(\sum_{i=1}^{n+1} X_i W_i\right) \tag{3.1.3}$$

具体算法如下：

(1)初始化. 给 $W_i(0)$各赋一个较小的随机非零值. 这里 $W_i(t)$为 t 时刻第 i 个输入的权值 $(1\leqslant i\leqslant n)$，$W_{n+1}(t)$为 t 时刻的阈值.

(2)输入样本. $X=(X_1,X_2,\cdots,X_n,T)$，T 称为教师信号，在两类样本分类中，如果 $X\in A$ 类，则 $T=1$；如果 $X\in B$ 类，则 $T=-1$.

(3)计算实际输出，

$$Y(t) = f\left(\sum_{i=1}^{n+1} X_i W_i(t)\right) \tag{3.1.4}$$

(4)修正权值，

$$W_i(t+1) = W_i(t) + \eta(T - Y(t))X_i, \quad i = (1,2,\cdots,n,n+1) \tag{3.1.5}$$

其中，$0<\eta\leqslant 1$，用于控制修正速度，通常不能太大，否则会影响 $W_i(t)$的稳定，也不能太小，太小会使 $W_i(t)$的收敛速度太慢.

(5)转到(2)，直到 W 对一切样本均稳定不变为止.

用单层感知器可实现部分逻辑函数，如：

$X_1 \wedge X_2$：$Y = 1\cdot X_1 + 1\cdot X_2 - 2$　即　$W_1 = W_2 = 1, \theta = 2$

$X_1 \vee X_2$：$Y = 1\cdot X_1 + 1\cdot X_2 - 0.5$　即　$W_1 = W_2 = 1, \theta = 0.5$

$\overline{X}_1$：$Y = (-1)X_1 + 0.5$　即　$W_1 = -1, \theta = -0.5$

三、单层感知器的局限性

1969 年 Minsky 和 Papert 出版了 Perception 一书，他们从数学上分析了以单层感知器为代表的人工神经网络系统的功能和局限性，指出单层感知器仅能解决一阶谓词逻辑和线性分类问题，不能解决高阶谓词和非线性分类问题. 为解决高阶谓词和非线性分类问题，必须引人含有隐层单元的多层感知器. 他们举出了异或(XOR)问题不能使用单层感知器来解决. 异或逻辑(见表 3.1)为 $X_1\overline{X}_2 \vee \overline{X}_1 X_2$，假定单层感知器能实现异或逻辑，那么，$Y=W_1X_1+W_2X_2-\theta$，要求：

$$W_1 + W_2 - \theta < 0 \rightarrow W_1 + W_2 < \theta$$

$$0 + 0 - \theta < 0 \rightarrow 0 < \theta$$

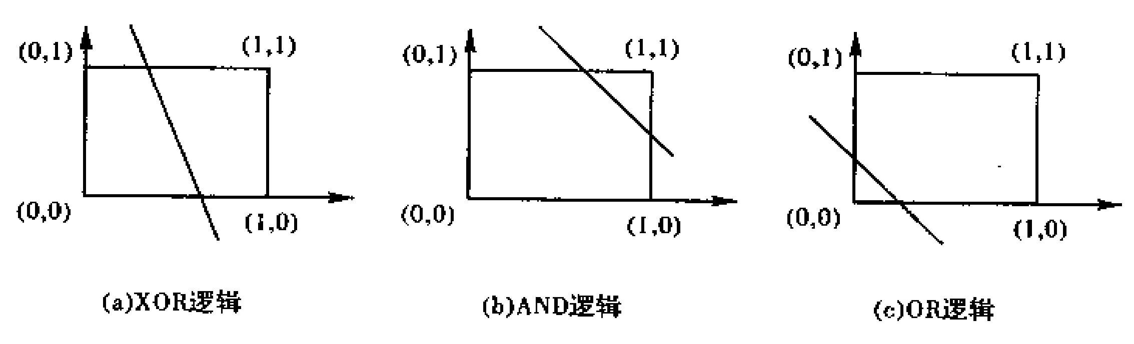
$$W_1 + 0 - \theta \geqslant 0 \rightarrow W_1 \geqslant \theta$$

$$0 + W_2 - \theta \geq 0 \rightarrow W_2 \geq \theta$$

表 3.1　异或逻辑

输入样本	输出
00	0
01	1
10	1
11	0

显然上述方程组无解，即感知器不能解决异或问题. 下面从几何图形来说明感知器的分类问题. 异或问题的几何意义可看做是笛卡儿平面上的一个正方形，其顶点坐标分别为(0,0)、(1,0)、(0,1)、(1,1). 如图 3.3(a)所示，不存在一条直线能将 X_1 XOR $X_2=1$ 与 X_1 XOR $X_2=0$ 的点分开. 但是从图 3.3(b)和 3.3(c)可以看出，对于 AND 和 OR 逻辑很容易找到一条直线将两类输出分在直线的两侧.



(a)XOR逻辑　　(b)AND逻辑　　(c)OR逻辑

图 3.3　线性可分性

3.1.2　感知器的收敛定理

一、线性可分函数

对给定的 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$，若存在 W，θ 和线性映像函数 f，使得

$$f: \mathbf{R}^n \rightarrow \{1, -1\}, X \in \mathbf{R}^n$$

则称 f 为线性可分函数.

所谓的线性可分是指存在一个超平面(二维为一条直线)能将两类样本分开.

对于上面的异或逻辑可用一个平面将其输出类别分开. 平面方程为：

$$X_1W_1 + X_2W_2 + X_3W_3 = \theta$$

$$X_1W_1 + X_2W_2 + (X_1 \wedge X_2)W_3 = \theta$$

其输入输出关系，见表 3.2.

表 3.2　三维异或逻辑

输入样本	输出
000	0
010	1
100	1
111	0

从表 3.2 可以看出前两维为异或，后一维是前两维的“与”. 图 3.4 给出了其三维表示.

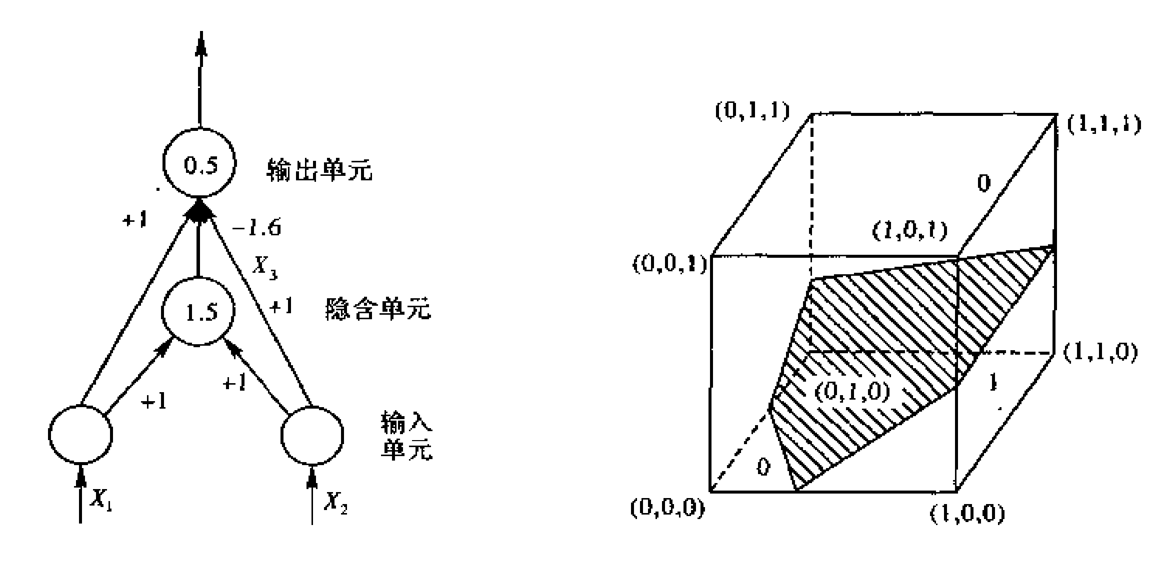


图 3.4 异或问题的三维表示

二、感知器收敛定理

定理 3.1 若函数 f 是线性可分的,则感知器的学习算法在有限次叠代后收敛.为证明此定理,先做一些简化:

(1)令 $\|X_k\|=1$(即学习样本都是单位向量);

(2)若 $Y_0<0$,则用 $-X_k$ 代替 X_k,因而对所有的 k,都有 $Y_k>0$(因 f 是线性可分的).

这样,要证明上述定理只要证明以下的结论即可.

因为 k 个样本是线性可分的,若存在一个 W^*,对所有的样本 k 使得 $W^* \cdot X_k>\delta$ 都成立,$\delta>0$.则下面步骤中的第④步仅需有限次.

①置 $t=1$,选初值 $W(t)$ 为不等于 0 的值;

②任选 $k\in\{1,N\}$,置 $X(t)=X_k$;

③若 $W(t)\cdot X(t)\geqslant 0$ 返回②,否则

④令 $W(t+1)=W(t)+X(t)$,$t=t+1$,返回②.

证明 $C(t)$ 表示向量 $W(t)$ 与 W^* 间夹角余弦,即

$$C(t)=\frac{W^*\cdot W(t)}{\|W(t)\|} \tag{3.1.6}$$

$$W^*\cdot W(t+1)=W^*\cdot[W(t)+X(t)]=W^*\cdot W(t)+W^*\cdot X(t)\geqslant X^*\cdot W(t)+\delta$$

所以

$$W^*\cdot W(t)\geqslant t\delta$$

$$\|W(t+1)\|^2=\|W(t)\|^2+2W(t)\cdot X(t)+\|X(t)\|^2Z<\|W(t)\|^2+1$$

所以

$$\|W(t)\|^2<t, C(t)>\frac{t\delta}{\sqrt{t}}$$

因为 $C(t)<0$,所以 $t\leqslant\frac{1}{\delta^2}$ 为一有限数.

证毕.

3.1.3 多层感知器网络

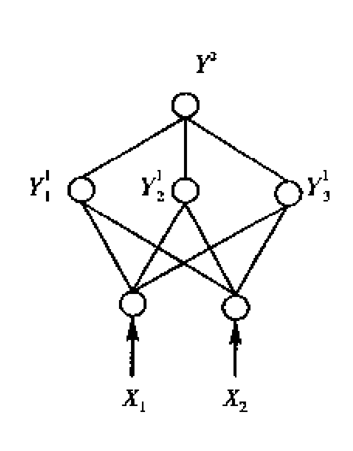
一、多层感知器网络

一个有两个隐层感知器的结构见图 3.5(b)所示，其中输入层有 n 个节点，第一隐层有 n_1 个节点，第二隐层有 n_2 个节点，各层节点的输出为：

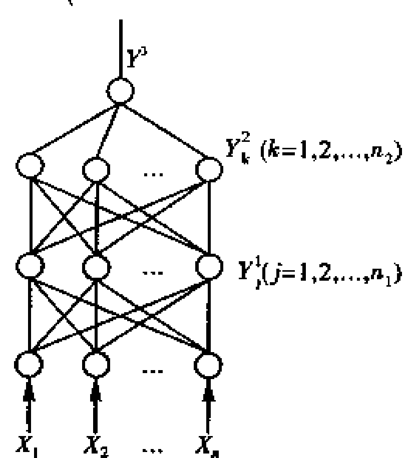
$$Y_j^1 = f\left(\sum_{i=1}^{n} W_{ij} \cdot X_i - \theta_j\right), \quad j = 1,2,\cdots,n_1 \tag{3.1.7}$$

$$Y_k^2 = f\left(\sum_{j=1}^{n_1} W_{jk} \cdot Y_j^1 - \theta_k\right), \quad j = 1,2,\cdots,n_2 \tag{3.1.8}$$

$$Y^3 = f\left(\sum_{k=1}^{n_2} W_k \cdot Y_k^2 - \theta\right) = f(\text{net}) = \begin{cases} 1 & \text{net} \geq 0 \\ -1 & \text{net} < 0 \end{cases} \tag{3.1.9}$$



(a)一个隐层的感知器



(b)两个隐层的感知器

图 3.5　多层感知器网络

二、多层感知器的分类决策能力

对于多层感知器网络，有以下定理：

定理 3.2　假定隐层的节点可以根据需要自由设置，那么用三层的阈值网络可以实现任意的二值逻辑函数.

首先，讨论一个隐层的情况，图 3.5(a)中输出层节点的输出为：

$$Y^2 = f\left(\sum_{j=1}^{n_1} W_j \cdot Y_j^2 - \theta\right) \tag{3.1.10}$$

此时隐层与 n 个输入节点的关系如同单层感知器一样，可以形成 n_1 个 n 维空间的超平面把 n 维输入空间分成一些小的于空间. 例如，$n=2$，$n_1=3$ 的情况下，隐层第 j 个节点的输出为：

$$Y_j^2 = f\left(\sum_{i=1}^{n} W_{ij} \cdot X_i - \theta_j\right), \quad j = (1,2,3) \tag{3.1.11}$$

可以在二维输入空间上决定三条直线，因为各自的 W_{ij} 和 θ_j 不同，三条直线的截距和斜率各不相同，如图 3.6(a)所示，就可以找到一个区域使其内为 A 类，之外为 B 类，用这三个隐单元所得到的一个封闭区域就可满足条件. 从隐单元到输出层只要满足下式即可得到正确划分.

$$Y^2=\{(X_1,X_2)[(W_{11}\cdot X_1+W_{21}\cdot X_2-\theta_1)>0\cap(W_{12}\cdot X_1+W_{22}\cdot X_2-\theta_2)>0$$
$$\cap(W_{13}\cdot X_1+W_{23}\cdot X_2-\theta_3)>0]\} \tag{3.1.12}$$

十分明显，隐节点到输出节点之间为“与”关系．对于图 3.6(b)，可以采用有两个隐层的感知器来实现，其中第二隐层节点到输出层节点为“或”关系，即满足下式即可．

$$Y^3=\{(X_1,X_2)[Y_1^2\cup Y_2^2]\}=\{(X_1,X_2)[(\bigcap_{j=1}^{4}(W_{1j}\cdot X_1+W_{2j}\cdot X_2-\theta_j)>0)$$
$$\cup(\bigcap_{j=4}^{6}(W_{1j}\cdot X_1+W_{2j}\cdot X_2-\theta_j)>0)]\} \tag{3.1.13}$$

(a)　　(b)

图 3.6　多层感知器对输入空间的划分

图 3.7 为一个解决异或问题的三层感知器网络，求得的权值及节点输入输出关系见下式．

$$Y_1^1=1\cdot X_1+1\cdot X_2-1$$
$$Y_2^1=(-1)\cdot X_1+(-1)\cdot X_2-(-1.5)$$
$$Y_1^2=1\cdot Y_1^1+1\cdot Y_2^1-2$$

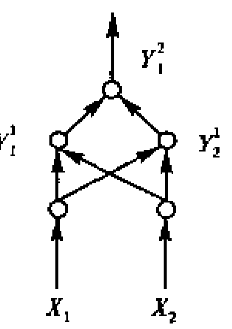


图 3.7　解决异或问题的三层感知器

三层感知器可以识别任一凸多边形或无界的凸区域．因为第一层可以分开两个半平面，第二层单元对第一层单元的输出作逻辑与运算，则它的输出即为多个半平面的交集构成的凸多边形．更多层可以实现更为复杂的分类，见图 3.8.

结构	判别域	异或	复杂区域分类	一般判别域形状
单层	半平面	A B B A	B A	
双层	开放式闭合的凸区	A B B A	B A	
三层	任意复杂形状的域	A B A B	B A	

图 3.8　单层与多层感知器的决策分类能力

从本节讨论中可知，对于单层感知器仅能解决线性可分问题，二维输入向量可以确定一条直线，三维输入向量可以确定一个平面，多维输入向量可以确定一个超平面将两类分开. 多层感知器可以解决线性不可分的问题，二维输入向量可以确定一个或几个凸、凹多边形，三维输入向量可以确定一个或几个凸、凹立方体，多维输入向量可以确定一个或几个超凸、凹立方体将两类分开. 对于多层感知器的学习训练，隐层和隐层间以及隐层和输出层间依据“与”、“或”的关系来确定权值和阈值.

3.1.4 感知器用于分类问题的算例

感知器的结构见图 3.9 所示. 其中，$u=W_1X_1+W_2X_2$，在此特选定输出单元为非线性函数，其输出为：

$$Y=f(u)=\frac{1}{1+e^{\frac{-2u}{u_0}}} \tag{3.1.14}$$

图 3.9 感知器结构

输入模式为：

(0.5, 0.05)、(0.05, 0.5)→A 类

(0.95, 0.5)、(0.5, 0.95)→B 类

教师信号为：

$$T=\begin{cases}1\text{:分类 A}\\0\text{:分类 B}\end{cases}$$

感知器的训练学习过程为不断地调节 W 和 θ 的过程，使得输出 Y 和教师信号 T 的误差小于规定的上限值，即告结束.

$$W_1(t+1)=W_1(t)+\alpha(T-Y)X_1$$

$$W_2(t+1)=W_2(t)+\alpha(T-Y)X_2$$

$$\theta(t+1)=\theta(t)+\beta(T-Y)$$

总的误差之和为：

$$E=\sum_{i=1}^{4}|T_i-Y_i| \tag{3.1.15}$$

模拟实现的程序框图见图 3.10. 表 3.3(a)和(b)分别为程序的输入参数和对于四个样本的输出结果. 图 3.11(a)为学习次数和误差关系曲线，(b)表示从初始状态($t=0$)到学习次数每增加 50 次时，分开 A 类和 B 类的直线的变化情况，随着学习的进行，逐渐逼近通过(1,0)和(0,1)的直线，学习终了时：$W_1=-0.97$，$W_2=0.97$，$\theta=0.97$，直线为：$W_1X_1+W_2X_2+\theta=0$.

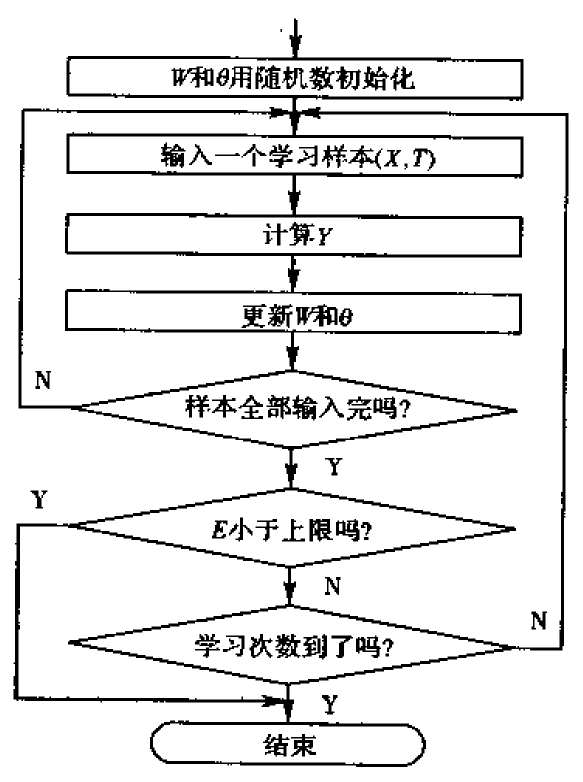


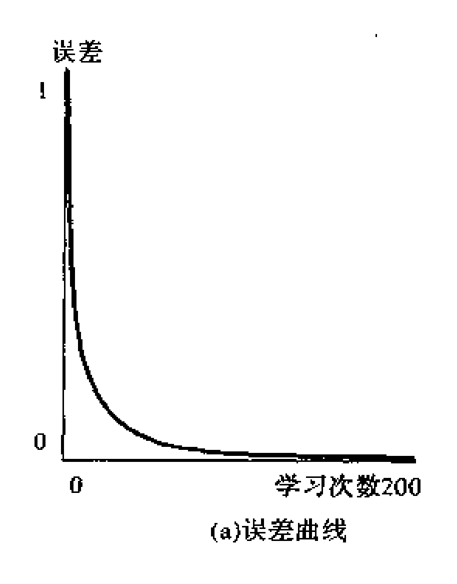
图 3.10 程序框图

表 3.3 (a)

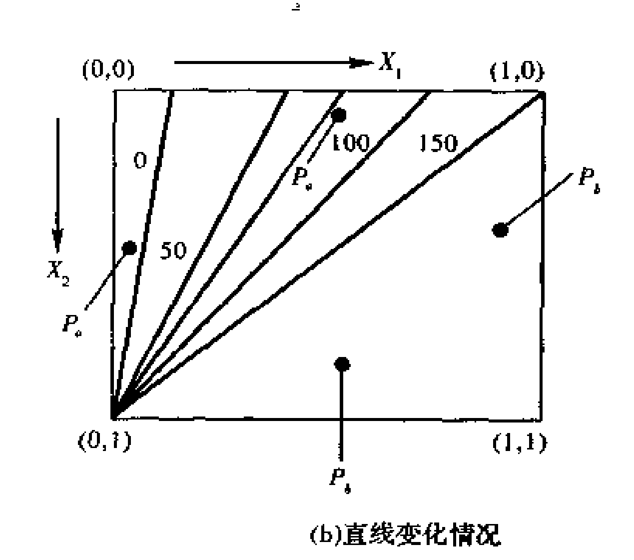
参　数	取　值
W 随机范围	0.20
θ随机范围	0.10
u_0	0.20
α	0.40
β	0.30
误差上限	0.01
最大学习次数	200

表 3.3 (b)

X_1	X_2	Y
0.50	0.05	0.99
0.05	0.50	0.99
0.95	0.50	0.01
0.50	0.95	0.01



(a)误差曲线



(b)直线变化情况

图 3.11

§3.2　多层前馈型神经网络

3.2.1　网络结构及工作过程

前馈型(feedforward type)神经网络主要实现非线性映射功能，还具有推广与概括能力，即当输入模式不是学习样本时，网络也能给出与该输入最接近的输入样本对应的期望输出.

一、学习样本

输入样本为：(X_K, T_K)，其中 $K \in \{1,2,\cdots,N\}$，N 为学习样本数，$X_K \in \mathbf{R}^n$，$T_K \in \mathbf{R}^m$.

二、工作过程

有两个隐层的网络结构见图 3.12 所示. 将学习样本(X_k, T_K)送入输入层，输入层不做任何处理而以全互联的方式传给第一隐层，第一隐层对输入信号经加权求和后，依据转移函

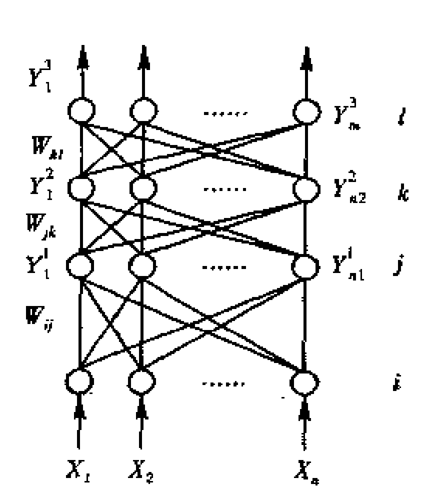


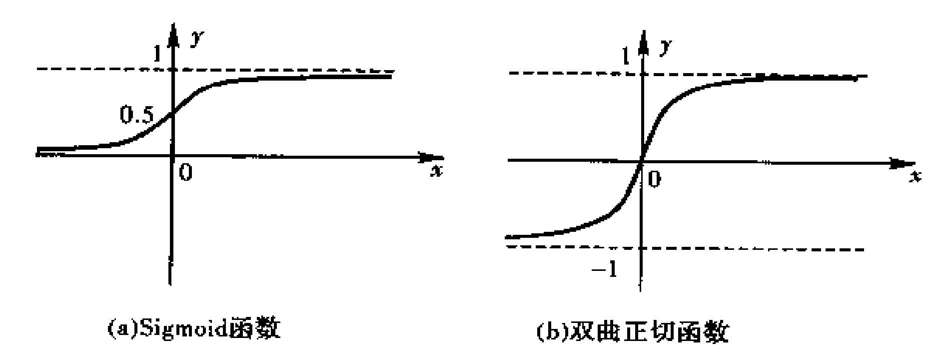
图 3.12　前馈型神经网络结构

数进行处理，其输出又以全互联的方式传给第二隐层，直至输出层输出向量 Y_k，其中，$Y_k \in \mathbf{R}^m$.

$$
\begin{cases}
Y_l^3 = f\left(\sum_{k=1}^{n_2} W_{kl} \cdot Y_k^2 - \theta_l\right) \\
Y_k^2 = f\left(\sum_{j=1}^{n_1} W_{jk} \cdot Y_j^1 - \theta_k\right) \\
Y_j^1 = f\left(\sum_{i=1}^{n} W_{ij} \cdot X_i - \theta_j\right)
\end{cases}
$$

三、非线性单元常采用的转移函数

神经元的转移函数用得最多的是 Sigmoid 函数和双曲正切函数，它们的函数形状见图 3.13 所示.



(a)Sigmoid函数　(b)双曲正切函数

图 3.13　常用的转移函数

Sigmoid 函数的表达式为：

$$f(x) = \frac{1}{1 + \exp(-x)} \quad (0 < f(x) < 1) \tag{3.2.1}$$

通常用增加参数 α 和 θ 来调整函数的斜率和使其左右平移，

$$f(x) = \frac{1}{1 + \exp(-\alpha(x - \theta))} \tag{3.2.2}$$

Sigmoid 函数为一单调递增连续函数，且处处可导，其导数为：

$$f'(x) = f(x)(1 - f(x)) \tag{3.2.3}$$

Sigmoid 函数通过下式能够映射到(−1,1)范围：

$$f(x) = \frac{2}{1 + \exp(-x)} - 1 \tag{3.2.4}$$

$$f'(x) = \frac{1}{2}(1 - f(x))(1 + f(x)) \tag{3.2.5}$$

双曲正切函数的表达式为：

$$f(x) = \tanh(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \quad (-1 < f(x) < 1) \tag{3.2.6}$$

通常用增加参数 α 和 θ 来调整函数的斜率和使其左右平移，

$$f(x) = \frac{1 - \exp(-\alpha(x - \theta))}{1 + \exp(-\alpha(x - \theta))} \tag{3.2.7}$$

General Motors 研究室的 Surender K Kenue 提出了一种比上述两种函数收敛速度快的一组转移函数及其导数，见表 3.4 所示.

表 3.4 新的转移函数极其导数

	新的转移函数	一阶导数
1	$f_1(x)=\frac{2}{\pi}\arctan(\sinh(x))$	$f_1'(x)=\frac{2}{\pi}\mathrm{sech}(x)$
2	$f_2(x)=\tanh(x)$	$f_2'(x)=\mathrm{sech}^2(x)$
3	$f_3(x)=\frac{2}{\pi}\left[\frac{\tanh(x)}{\cosh(x)}+\arctan(\sinh(x))\right]$	$f_3'(x)=\frac{4}{\pi}\mathrm{sech}^3(x)$
4	$f_4(x)=\frac{3}{2}\tanh(x)\left[1-\frac{\tanh^2}{3}(x)\right]$	$f_4'(x)=\frac{3}{2\pi}\mathrm{sech}^4(x)$
5	$f_5(x)=\frac{2}{3\pi}\left[5\frac{\tanh(x)}{\cosh(x)}+3\arctan(\sinh(x))-2\frac{\tanh^3(x)}{\cosh(x)}\right]$	$f_5'(x)=\frac{16}{3\pi}\mathrm{sech}^5(x)$
6	$f_6(x)=\frac{15}{8}\left[\tanh(x)-\frac{2\tanh^3(x)}{3}+\frac{\tanh^5(x)}{5}\right]$	$f_6'(x)=\frac{15}{8\pi}\mathrm{sech}^6(x)$

3.2.2 误差函数与误差曲面

一、误差函数

误差函数：$E(\boldsymbol{W})=g(f(\boldsymbol{W},\boldsymbol{X}_k,\boldsymbol{T}_k))$，$k=1,2,\cdots,N$，$E$ 称为误差(测度)函数，即网络的实际输出向量 $\boldsymbol{Y}_k$ 与教师信号向量 $\boldsymbol{T}_k$ 的误差用误差函数来判别，常采用二乘误差函数加以判别，其中，N 为输入样本的个数，m 为输出向量的维数.

$$E=\frac{1}{2}\sum_{k=1}^{N}E_k=\frac{1}{2}\sum_{k=1}^{N}\|\boldsymbol{Y}_k-\boldsymbol{T}_k\|^2=\frac{1}{2}\sum_{k=1}^{N}\sum_{i=1}^{m}(Y_{ki}-T_{ki})^2 \tag{3.2.8}$$

二、映射

对于给定的一组数据$(\boldsymbol{X}_k,\boldsymbol{T}_k)(k=1,2,\cdots,N)$，网络一组特定的权值 W 实现一定精度的映射. 训练的目的是希望得到的权值能产生最小的误差和最好的精度.

把从$\{\boldsymbol{X}_k\}\subset\mathbf{R}^n$ 到$\{\boldsymbol{Y}_k\}\subset\mathbf{R}^m$ 的映射记为：

$$f:\boldsymbol{X}_k\subset\mathbf{R}^n\rightarrow\boldsymbol{Y}_k\subset\mathbf{R}^m$$

三、误差曲面

误差曲面：若隐层与输出层间的权值数记为 $m_n\cdot n_2$，对于给定的训练样本$(\boldsymbol{X}_k,\boldsymbol{T}_k)$，网络权矢量 $W(W_1,W_2,\cdots,W_{m\cdot n_2})$通过误差函数所计算出来的映射误差所描绘出的曲面. 误差曲面可用$(m\cdot n_2+1)$维空间来描述，即是$(m\cdot n_2+1)$空间的一个曲面. 不同的 $E(\boldsymbol{W})$有不同的误差曲面形状.

网络学习：是指按照某种学习规则选取新的 $\boldsymbol{W}'$，使得 $E(\boldsymbol{W}')\leqslant E(\boldsymbol{W})$，对于误差曲面上的点 $E(\boldsymbol{W})$总是向山下移动，最终移到最深的谷底(全局最小). 若曲面有多个谷底，移动的过程可能陷入局部极小.

移动步长：也称学习率，步长小移动轨迹较平滑、慢、易陷入局部极小；步长大，速度快，可能跳过局部极小，也可能跳过全局最小点，也易产生振荡. 一般情况下，开始时步长大，后期步长小.

梯度下降算法：如果移动是在误差曲面最陡的方向或梯度下降的方向进行，这样下山的速度快，称作最速梯度下降法.

3.2.3 网络的学习规则——梯度下降算法

权值的修正量取误差函数 $E(W)$ 对 W 的负梯度，即：

$$\boldsymbol{W}(t+1)=\boldsymbol{W}(t)+\Delta\boldsymbol{W}(t) \tag{3.2.9}$$

$$\Delta\boldsymbol{W}(t)=-\eta\frac{\partial E(\boldsymbol{W})}{\partial(\boldsymbol{W})} \tag{3.2.10}$$

设有 N 个学习样本 $(\boldsymbol{X}_k,\boldsymbol{T}_k)(k=1,2,\cdots,N)$，对于某个 $\boldsymbol{X}_k$ 网络输出为 $\boldsymbol{Y}_k$，节点 i 的输出为 Y_{ik}，i 和 j 的连接权值为 W_{ij}，节点 j 的输入加权和为：

$$\mathrm{net}_{jk}=\sum_i W_{ij}Y_{ik} \tag{3.2.11}$$

误差函数使用二乘误差函数：

$$E=\frac{1}{2}\sum_{k=1}^{N}\|\boldsymbol{T}_k-\boldsymbol{Y}_k\|^2 \tag{3.2.12}$$

$$E_k=\|\boldsymbol{T}_k-Y_k\|^2=\sum_{j,k}(T_{jk}-Y_{jk})^2 \tag{3.2.13}$$

定义：

$$\delta_{jk}=\frac{\partial E_k}{\partial \mathrm{net}_{jk}} \tag{3.2.14}$$

$$O_{jk}=f(\mathrm{net}_{jk}) \tag{3.2.15}$$

$$\frac{\partial E_k}{\partial W_{ij}}=\frac{\partial E_k}{\partial \mathrm{net}_{jk}}\frac{\partial \mathrm{net}_{jk}}{\partial W_{ij}}=\frac{\partial E_k}{\partial \mathrm{net}_{jk}}Q_{ik}=\delta_{jk}O_{ik} \tag{3.2.16}$$

(1)当 j 为输出节点时：

$$O_{jk}=Y_{jk}$$

$$\delta_{jk}=\frac{\partial E_k}{\partial \boldsymbol{Y}_{jk}}\frac{\partial Y_{jk}}{\partial \mathrm{net}_{jk}}=-2(T_{jk}-Y_{jk})f'(\mathrm{net}_{jk}) \tag{3.2.17}$$

(2)若 j 不是输出节点时，有：

$$\delta_{jk}=\frac{\partial E_k}{\partial \mathrm{net}_{jk}}=\frac{\partial E_k}{\partial O_{jk}}\frac{\partial O_{jk}}{\partial \mathrm{net}_{jk}}=\frac{\partial E_k}{\partial O_{jk}}f'(\mathrm{net}_{jk}) \tag{3.2.18}$$

$$\frac{\partial E_k}{\partial O_{jk}}=\sum_m\frac{\partial E_k}{\partial \mathrm{net}_{mk}}\frac{\partial \mathrm{net}_{mk}}{\partial O_{jk}}=\sum_m\frac{\partial E_k}{\partial \mathrm{net}_{mk}}\frac{\partial(\sum_j W_{jm}O_{jk})}{\partial O_{jk}}=\sum_m\delta_{mk}W_{jm} \tag{3.2.19}$$

$$\delta_{jk}=f'(\mathrm{net}_{jk})\sum_m\delta_{mk}W_{jm} \tag{3.2.20}$$

因而对权值的修正为：

$$W_{ij}(t+1)=W_{ij}(t)-\eta\sum_{k=1}^{N}\frac{\partial E_k}{\partial W_{ij}}=W_{ij}(t)-\eta\sum_{k=1}^{N}\delta_{jk}O_{ik} \tag{3.2.21}$$

§ 3.3 误差逆传播算法(BP 算法)

1985 年，Rumelhart，Hinton 和 Williams 提出了用于前向神经网络学习训练的误差逆传播算法(Back Propagation，简称 BP 算法)，成功地解决了多层网络中隐含层神经元连接权值的学习问题。

3.3.1 BP 算法的数学描述

BP 算法是有教师指导的，适合于多层神经网络的学习训练，是建立在梯度下降算法基础上的. 主要思想是把学习过程分为两个阶段，第一阶段（信号正向传播过程），输入信号通过输入层经隐层逐层处理并计算每个节点的实际输出值；第二阶段（误差修正反向传播过程），若在输出层未得到期望的输出值，则逐层递归地计算实际输出与期望输出之间的误差，并依据此误差来修正权值.

在学习过程中，对于每一个输入样本逐次修正权值向量，若有 N 个输入样本，那么，一次学习过程将对权值向量修正 N 次，这种逐次不断修正权值向量的方法称之为逐次修正法.

一、确定和选择网络结构

包括确定输入层和输出层的节点数，选择隐层数和各隐层内的节点数. 确定节点的转移函数、误差函数类型和选择各个可调参数值.

基于 BP 算法的一个三层前向神经网络的结构见图 3.14 所示.

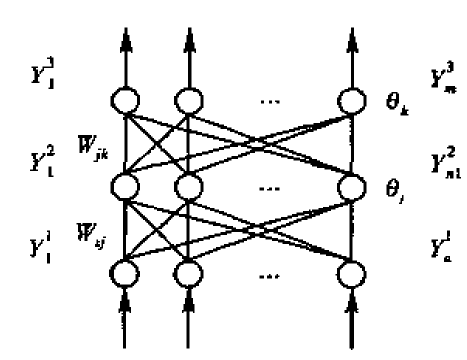


图 3.14　一个三层的前向神经网络

图中：Y_i^1 为输入层节点 i 的输出；Y_j^2 为中间层节点 j 的输出；Y_k^3 为输出层节点 k 的输出；T_k 为输出层节点 k 对应的教师信号；W_{ij} 为节点 i 和节点 j 间的连接权值；W_{jk} 为节点 j 和节点 k 间的连接权值；θ_j 为中间层节点 j 的阈值；θ_k 为输出层节点 k 的阈值.

神经元的转移函数选择为 Sigmoid 函数：

$$f(x)=\frac{1}{1+\exp(-2x/u_0)} \tag{3.3.1}$$

$$f'(x)=\frac{2}{u_0}f(x)(1-f(x)) \tag{3.3.2}$$

误差函数为平方误差函数：

$$E=\frac{1}{2}\sum_{k=1}^{N}\|\mathbf{T}_k-\mathbf{Y}_k^3\|^2 \tag{3.3.3}$$

二、初始化

(1)设定学习次数 $t=0$，对网络权值和阈值赋于小的随机数，$W_{ij}(t)\in[-1,1]$，$W_{jk}(t)\in[-1,1]$，$\theta_j(t)\in[-1,1]$，$\theta_k(t)\in[-1,1]$.

1. 前向计算

(2)输入一个学习样本$(\mathbf{X}_k,T_k)$，其中 $k\in\{1,2,\cdots,N\}$，N 为样本数，$X_k\in\mathbf{R}^n$，$T_k\in\mathbf{R}^m$.

(3)计算隐层各节点的输出值：

$$Y_j^2=f\Big(\sum_{i=1}^{n_1}\mathbf{W}_{ij}Y_i^2-\theta_j\Big)=f\Big(\sum_{i=1}^{n}\mathbf{W}_{ij}\mathbf{X}_{Ki}-\theta_j\Big),\quad j\in\{1,2,\cdots,n_1\} \tag{3.3.4}$$

(4)计算输出层节点的输出：

$$Y_k^3=f\Big(\sum_{j=1}^{n_1}\mathbf{W}_{jk}Y_j^3-\theta_k\Big),\quad k\in\{1,2,\cdots,m\} \tag{3.3.5}$$

2. 逆向误差修正计算

(5)输出层节点误差的计算：

$$\delta_k = -(T_k - Y_k^3)Y_k^3(1 - Y_k^3),\quad k \in \{1,2,\cdots,m\} \tag{3.3.6}$$

(6)隐层节点误差的计算：

$$\delta_j = Y_j^2 \cdot (1 - Y_j^2)\sum_{k=1}^{m}\delta_k W_{jk} \tag{3.3.7}$$

(7)用(5)求出的误差修正量 δ_k 来修正输出层和隐层间连接权值矩阵 W_{jk} 和阈值向量 $\boldsymbol{\theta}_k$. 例如对节点 k 和隐层 j 的连接权值 W_{jk} 和节点 k 的阈值的修正为：

$$W_{jk}(t+1) = W_{jk}(t) + \alpha\,\delta_k Y_j^2 \tag{3.3.8}$$

$$\boldsymbol{\theta}_k(t+1) = \boldsymbol{\theta}_k(t) + \beta\,\delta_k \tag{3.3.9}$$

(8)用(6)求出的误差修正量 δ_j 来修正隐层和输入层间连接权值矩阵 W_{jk} 和阈值向量 $\boldsymbol{\theta}_j$. 例如隐层 j 和输入层节点 i 的连接权值 W_{ji} 和节点 j 的阈值的修正为：

$$W_{ij}(t+1) = W_{ij} + \alpha\,\delta_j Y_i^1 \tag{3.3.10}$$

$$\boldsymbol{\theta}_j(t+1) = \boldsymbol{\theta}_j(t) + \beta\,\delta_j \tag{3.3.11}$$

(9)如果全部学习样本未取完，则返回(2)，否则，

(10)计算误差函数 E，并判断 E 是否小于规定的误差上限，如果 E 小于误差上限，则算法结束；否则，如果学习次数到，则算法结束；否则更新学习次数 $t=t+1$，返回(2).

算法的流程见图 3.15.

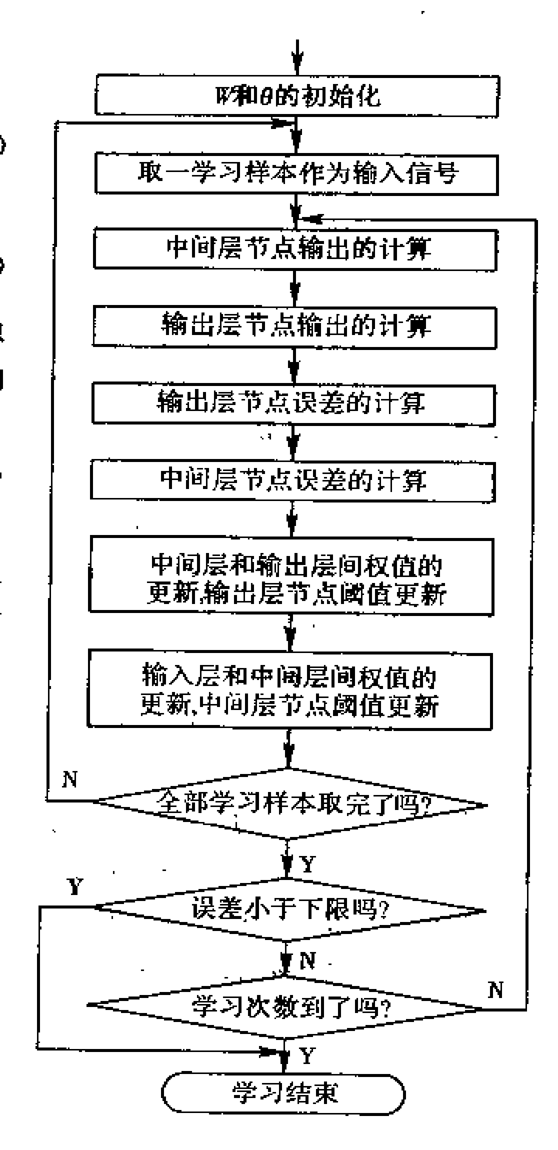


图 3.15　BP 算法程度流程图

上面的算法针对每个输入样本进行一次权值和阈值的修正，而一括修正法对每个输入样本都计算修正量，对权值修正量逐次累加，仅当全部学习样本学习结束后才真正地修正权值和阈值，被称为批处理的修正方法. 一括修正法可以减少修正次数，因而可以缩短学习时间，但是由于修正量被平均化，不能进行细微的修正. 对于一些复杂问题，特别是当学习样本数多的时候可能得不到好的效果.

一括修正法对图 3.15 的变更部分见图 3.16 所示.

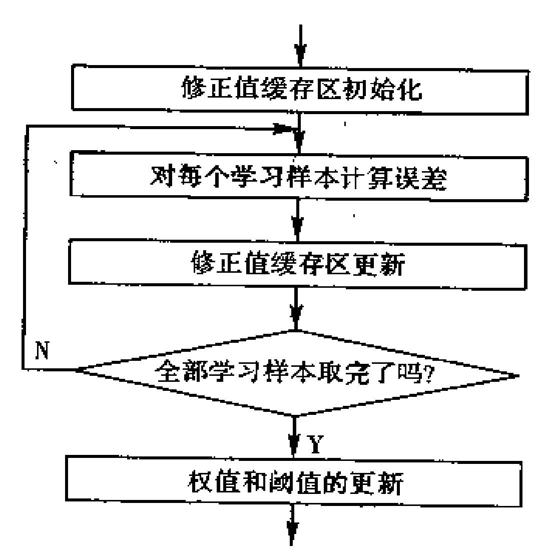


图 3.16　一括修正部分程序流程图

使用 Memond 法修正权值向量和阈值向量时，要考虑到前一次修正量. 如果 $(t-1)$ 时刻的修正量为 $\Delta W(t-1)$，t 时刻计算的修正量为 $\Delta W(t)$，设 Memond 系数为 m，则 Memond 法对权值的修正量为：

$$\Delta W(t+1) = \Delta W(t) + m\Delta W(t-1) \tag{3.3.12}$$

当 t 时刻计算的修正量为 $\Delta W(t)$ 和 Memond 项 $\Delta W(t-1)$ 符号相异时能使本次的修正量 $\Delta W(t)$ 值变

小，即能抑制振荡，因而能加快学习过程．为使本次修正量更接近于前一次的修正方向，应该使 Memond 系数不断增加，修改其修正量为：

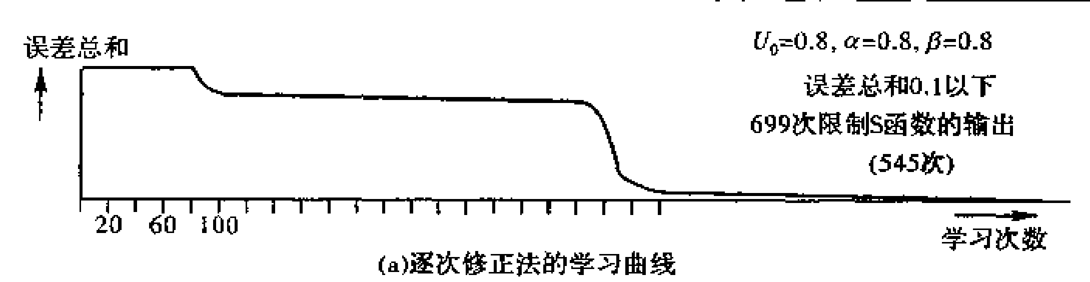
$$\Delta W(t+1) = \Delta W(t) + m(t)\Delta W(t-1) \tag{3.3.13}$$

$$m(t) = \Delta m + m(t-1) \tag{3.3.14}$$

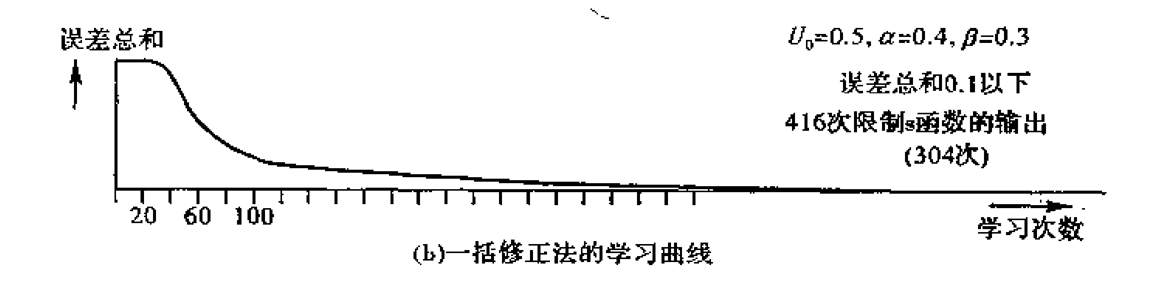
各种误差修正方法在学习过程中的误差曲线和学习次数见图 3.17 所示．Sigmoid 函数参数 u_0 和学习次数的关系见表 3.5.

表 3.5　Sigmoid 函数参数 u_0 对算法学习次数的影响

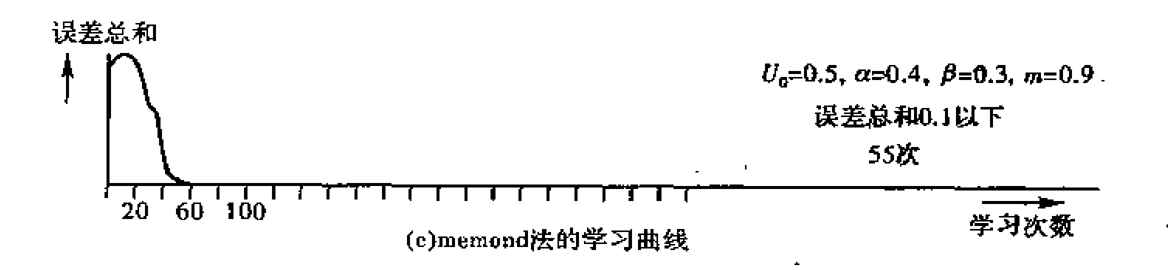
参数 u_0	Memond 法 $m=0.9$	修正 memond 法 $m=0.6, \Delta m=0.02$
0.5	55	31
0.6	47	43
0.7	63	61



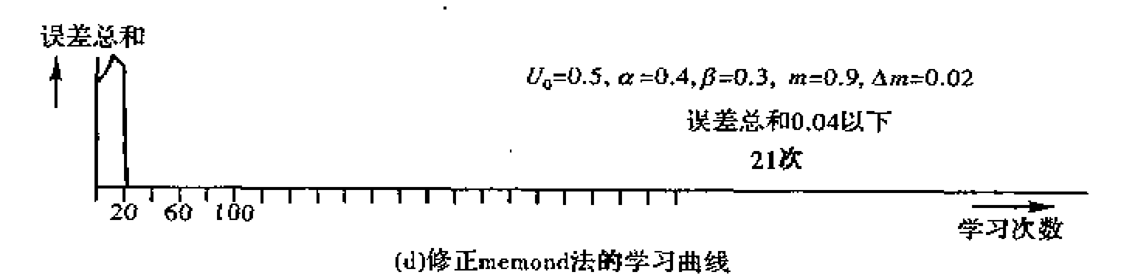
(a)逐次修正法的学习曲线



(b)一括修正法的学习曲线



(c)memond法的学习曲线



(d)修正memond法的学习曲线

图 3.17　各种修正权值方法的比较

3.3.2 BP算法收敛性定理

算法把一组样本I/O的问题变为一个非线性优化问题，使用了优化中最普通的梯度下降算法，用迭代运算求解权值相应于学习记忆，加入隐节点使优化问题的可调参数增加，从而可以得到更精确的解.

误差逆传播算法只要通过有限次迭代，就能保证学习收敛. 下面的定理，在数学上给予保证.

定理 3.3 令 $\Phi(X)$ 为一有界单调递增连续函数，K 为 $\mathbf{R}^n$ 维的有界闭集合，$f(X)=f(X_1,X_2,\cdots,X_n)$ 是 K 上的连续函数，那么对于任意的 $\varepsilon>0$，存在正整数 N 和常数 C_i、$\theta_i(i=1,2,\cdots,N)$ 和 $W_{ij}(i=1,2,\cdots,N;j=1,2,\cdots,n)$ 使

$$\max_{X\in K}\left|f(X_1,X_2,\cdots,X_n)-\sum_{i=1}^{N}C_i\Phi\left(\sum_{j=1}^{n}W_{ij}\cdot X_j-\theta_i\right)\right|<\varepsilon \tag{3.3.15}$$

成立.

此定理说明对于任意 $\varepsilon>0$，存在一个三层网络，其隐单元的输出函数为 $\Phi(X)$，输出单元为线性的，对于任意连续映射 $f:\mathbf{R}^n\to\mathbf{R}^m$，在任意的有界闭集合上能以任意精度逼近.

BP算法虽然简单，对各个方面都有重要意义，但是它存在以下问题：

①从数学上看，它是一个非线性优化的问题，这就不可避免地存在局部极小的问题.

②学习算法的收敛速度很慢，通常需要几千步迭代或更多.

③网络的运行还是单向传播，没有反馈，目前这种模型并不是一个非线性动力学系统，只是一个非线性映射.

④网络的隐节点数目选取尚无理论上的指导，而是根据经验或实验选取.

⑤对于新加入的样本要影响已经学完的样本，不能在线学习，同时描述每一个样本的特征数目也要求必须相同.

§3.4 误差逆传播算法(BP算法)的若干改进

基于BP算法的前馈型网络，通过许多具有简单处理能力的神经元的复合作用使网络具有复杂的非线性映射能力，它不属于一个非线性动力学系统，只实现一个非线性映射. 尽管如此，由于它理论上的完整性和成功地应用于广泛的应用问题，所以它仍然有重要的意义，但BP算法也存在不少问题.

(1)已学习好的网络的推广(泛化)问题，即能否逼近规律和对于大量未经学习过的输入向量也能正确处理，并且网络是否具有一定的预测能力.

(2)基于BP算法的网络的误差曲面有三个特点：

第一，可能有很多全局最小的解，即多解问题.

第二，存在一些平坦区，在此区域内误差改变很小，这些平坦区多数发生在神经元的输出接近于0或1的情况下，对于不同的映射，其平坦区的位置、范围各不相同，在有些情况下，误差曲面会出现一些阶梯形状.

第三，存在许多局部极小点. 在某些初始条件下，算法的结果会陷入局部极小点. 初始随机权值的大小，对局部极小的影响极大，如果权值太大，一开始就可能使网络处于S型函数的饱

和区，则系统有可能陷入局部最小(或非常平坦区)．为防止这种现象，选取随机加权值时，使节点的输入加权和的绝对值小于 1，但又不能太小，为此可选择各加权值的最级为$1/\sqrt{k_i}$，其中 k_i 表示连到 j 点的所有前层结点的数目．一般来说，希望初始权值在输入累加时，使每个神经元的加权求和接近于零，这样可保证在一开始时不会落到那些平坦区上，保证每个神经元在一开始都在它们激活函数变化最大的地方进行．

(3)学习算法的收敛速度慢．

(4)网络的隐节点个数的选取尚缺少统一而完整的理论指导．

总的来说，隐含单元数与问题的要求、输入输出层节点数都有直接的关系．

3.4.1　基于全局学习速率自适应调整的 BP 算法

BP 算法优化的一种方法是对学习速率 η 自动地寻找合适的值．全局自适应调整算法是利用整个网络状态的知识(例如以前的权值调整方向)去修改全局参数，即对每个可调参数根据学习情况进行自适应调整．

1. 加入动量项

BP 基于算法的神经网络，在学习过程中，需要不断地改变权值，而权值的改变量是和误差相对于权值的导数成正比的．通常梯度下降方法的学习速率 η 是一个常数，而且 η 越大，权值的改变越大，若能选择合适的速率，使它的值尽可能的大，但又不至于引起振荡．即不断地修改学习速率，使它包含有一个动量项，就是在每次加权调节量上加上一项正比例于前次加权变化量的值(即本次权值的修改表达式中引入前次的权值修改量)．带有动量项的加权调节公式为：

$$\Delta w(t+1) = -\eta\frac{\partial E}{\partial w} + \alpha\Delta w(t) \tag{3.4.1}$$

其中，α 为动量系数，一般取 0.9 左右．

引入动量项的效果就是使得学习过程中等效地改变 η 值而不再是恒定的值．引入这个动量项之后，使得调节向着底部的平均方向变化，不致产生大的摆动，即动量起到缓冲平滑的作用．若系统进入误差曲面的平坦区，那么误差将变化根小，于是 $\Delta w(t+1)$ 近似等于$\Delta w(t)$，而平均的 Δw 将变为：

$$\Delta w \approx \frac{-\eta}{1-\alpha}\frac{\partial E}{\partial w} \tag{3.4.2}$$

式中$\frac{-\eta}{1-\alpha}$变化大，将调节尽快脱离饱和区．

再来看一下动量项的作用，以一个权变量 w_{ij} 为例，由式(3.4.1)改写为：

$$\Delta w_{ij}(t+1) = \Delta w'_{ij}(t+1) + \alpha\Delta w_{ij}(t) \tag{3.4.3}$$

其中 $\Delta w'_{ij}(t+1) = -\eta\partial E/\partial w_{ij}$，反映当前的方向．如果 $\Delta w'_{ij}(t+1)$ 与 $\Delta w_{ij}(t)$ 同号，那么表明，在误差超曲而上，在 w_{ij} 轴上沿这个方向在一定范围内误差有减小的趋势，因而可使 $\Delta w_{ij}(t+1)$ 修改量更大些，面 $\Delta w'_{ij}(t+1)$ 和 $\Delta w_{ij}(t)$ 正好能够实现这个要求；如果 $\Delta w'_{ij}(t+1)$ 与$\Delta w_{ij}(t)$ 符号相反，这表明在 w_{ij} 轴上，从 $\Delta w_{ij}(t)$ 到 $\Delta w'_{ij}(t+1)$ 已越过一个局部极小点，为避免振荡，希望 $\Delta w_{ij}(t+1)$ 修改量小些．而 $\Delta w'_{ij}(t+1)$ 与 $\alpha\Delta w_{ij}(t)$ 之和正满足了这个要求．综上所述，显然地看到动量项的引入加快了学习速度．

但是这种方法的缺点也是明显的，参数 α 的选取只能通过实验来确定．由式(3.4.1)可以

看出 α 起着决定过去权值的变化对当前权值变化的影响程度. 这样,就给出了权重空间中的一类动量,这类动量可以滤掉权重空间中误差曲面的高频偏差. 为了防止发散振荡,需要取很小的间隔,但这会使系统非常缓慢地移向深谷. 动量可以滤掉高曲率部分,并且可以使有效的权值间隔加大. 一般情况,工作中的大部分模拟所使用的 α 值大约为 0.9.

2. 学习速率的经验公式法

对于批处理更新的学习速率,是基于相类似训练模式产生类似梯度的假设. 如果有许多相类似的训练模式,期望减小学习速率,训练集能被分为 m 个相类似模式子集,学习速率和动量项用下面的方法设置.

$$\eta = 1.5/\sqrt{N_1^2 + N_2^2 + \cdots + N_m^2} \tag{3.4.4}$$
$$\alpha = 0.9$$

3. 学习速率渐小法

学习速率渐小法是在训练期间减小学习速率的方法. 这种方法称作为"Search-Then-Converge". 因为开始学习时,学习速率比较大,有利于加快学习速度,面快到极值点时,学习速率减小有利于收敛. 从大的学习速率 $\eta(0)$开始,在训练期间,这个值减小到大约$\frac{\eta(0)}{1+n}$,后来为

$$\eta(n) = \frac{\eta(0)}{1+n} \tag{3.4.5}$$

4. 渐进自适应学习速率

用一种简单的进化策略来调节学习速率. 从某个 V 开始,下一步更新通过用增加和减小学习速率去完成. 产生比较好性能中的一个被用作下一步更新的起始点:

①创建两个一样的网络和初始学习速率.

②按下式调节两个网络的权.

$$\Delta w_{ij}(n) = \frac{-\eta(n)\left(\frac{\partial E}{\partial w_{ij}}\right)}{\|\nabla E\|} \tag{3.4.6}$$

③如果两者总误差已经得到增加(回溯),放弃这些两络并重新启动以前的网络和初始学习速率.

④在减小总误差的情况下,用具有比较小的总误差的网络以及学习速率以启动下一个学习步.

3.4.2 基于局部学习速率自适应调整的 BP 算法

局部自适应指的是对每个可调节参数采用独自的学习速率,所以对每个权值寻找到最优学习速率. 局部策略用仅有的特定的信息(例如偏导数)去修改权值特定参数. 实践表明它是很有效的.

1. 基于符号变换的学习速率自适应

对每个速接采用单独的学习速率,这些学习速率的自适应是通过观察最后两个梯度的符号来完成的. 只要检测得在符号上不改变,相应的学习速率就增加,如果符号改变,学习速率就减小. 这是精确的算法. 工作步骤如下:

(1)对每个权重,选择某个小初值 $\eta_{ij}(0)$;

(2)修改学习速率

$$\eta_{ij}(n) = \eta_{ij}(n-1)u$$

如果

$$\frac{\partial E}{\partial w_{ij}}(n)\frac{\partial E}{\partial w_{ij}}(n-1) \geqslant 0, \quad \Delta w_{ij}(n) = -\eta_{ij}(n)\left(\frac{\partial E}{\partial w_{ij}} + \alpha \Delta w_{ij}(n-1)\right) \tag{3.4.7}$$

否则

$$\Delta w_{ij}(n) = -\eta_{ij}(n)\frac{\partial E}{\partial w_{ij}} + \alpha \Delta w_{ij}(n-1) \tag{3.4.8}$$

(3)更新连接. 只要保持 $u=1/d$,选择合适的参数 u 和 d 是很容易的. 推荐的值分别是1.1～1.3或者0.7～－0.9. 如果总误差增加. 用回溯策略重新启动更新步骤,对于这种重新启动,所有学习速率被减半.

2. Delta-Bar-Delta 技术

Delta-Bar-Delta 方法通过观察指数平均梯度的符号变化来控制学习速率. 通过加入常值代替乘这个值来提高学习速率:

(1)对每个权重,选择某个小的初值 $\eta_{ij}(0)$.

(2)修改学习速率,

$$\eta_{ij}(n) = \eta_{ij}(n-1) + u, \quad 如果\frac{\partial E}{\partial w_{ij}}(n)\delta_{ij}(n-1) > 0 \tag{3.4.9}$$

$$\eta_{ij}(n) = \eta_{ij}(n-1)d, \quad 如果\frac{\partial E}{\partial w_{ij}}(n)\delta_{ij}(n-1) < 0 \tag{3.4.10}$$

$$\eta_{ij}(n) = \eta_{ij}(n-1), \qquad 其他 \tag{3.4.11}$$

其中 $\delta(n)$ 表示指数平均梯度:

$$\delta_{ij}(n) = (1-\phi)\frac{\partial E(n)}{\partial w_{ij}} + \delta_{ij}(n-1) \tag{3.4.12}$$

(3)更新连接,

$$\Delta w_{ij}(n) = -\eta_{ij}(n)\frac{\partial E}{\partial w_{ij}} \tag{3.4.13}$$

对于 u 推荐很不同的值(5.0,0.095,0.085,0.035),对于 d,采用(0.9,0.85,0.666)和对于 ϕ 采用0.7. 特别是难于找到合适的 u,小的值可能产生慢自适应,而大的值危及学习过程.

3.4.3 BI(Back Impedance)算法

BI算法在进行权值调整时,不仅用到了 t 时刻的修正量,而且还用到 $t-1$ 和 $t-2$ 时刻的修正量. 这样在学习精度和学习时间上高于BP算法. 一些实验结果表明,特别是在函数逼近方面BI算法比BP算法在学习精度上高出一个数量级.

1. BI算法

使用BI算法的一个神经网络结构见图3.18所示. 有四层的前馈型网络,其中输入和输出层各有一个节点,两个隐层各有四个节点. BI学习算法如下:

①给权值赋予一个小的随机数.

②给定输入函数值与相应的输出函数值.

③计算每个节点的输出值,

$$Y_j = \frac{1}{1+\exp\left(-\sum W_{ij} \cdot Y_i\right)} \tag{3.4.14}$$

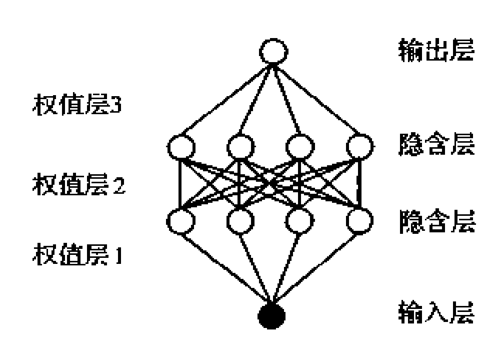


图 3.18　网络结构

○人工神经元　●输入端　——权值

④计算输出层节点的误差项，

$$\delta_l = (T_l - Y_l)Y_l(1 - Y_l) \tag{3.4.15}$$

$$\delta_k = Y_k(1 - Y_k)\sum_l \delta_l W_{kl} \tag{3.4.16}$$

$$\delta_j = Y_j(1 - Y_j)\sum_l \delta_k W_{jk} \tag{3.4.17}$$

⑤调整权值，

$$W_{ij}(t+1) = W_{ij}(t) + a\delta_j Y_i + b(W_{ij}(t-1)) + c(W_{ij}(t-1) - W_{ij}(t-2)) \tag{3.4.18}$$

式中，a 为学习率，相当于梯度下降算法中的学习步长；b 为影响从“前一次”权值改变到“当前”权值的权值空间运动方向，是影响权值变化的一个常数；c 为影响从“再前一次”权值改变到“前一次”权值的权值空间运动方向，也是影响权值变化的常数. a、b、c 三个常数满足下列关系，则收敛速度会加快：

$$a = \frac{1}{1+J+M+D} \tag{3.4.19}$$

$$b = \frac{2J+M}{J+M+D} \tag{3.4.20}$$

$$c = \frac{-J}{J+M+D} \tag{3.4.21}$$

式中 J、M、D 满足：

$$J\frac{\partial^3 W}{\partial t^3} + M\frac{\partial^2 W}{\partial t^2} + D\frac{\partial W}{\partial t} = -\frac{\partial E}{\partial W} \tag{3.4.22}$$

⑥给定另一输入函数值，返回②. 所有的输入函数值循环进行计算，直至所有权值稳定，网络误差达到预定精度算法结束.

2. BI 算法应用于函数非线性变换

网络的输入函数为：

$$V_i = \left\{A\left[\sum_{\delta=0.25}^{0.5,1.2}\frac{1}{\sqrt{2\delta}}\exp\left(\frac{-x^2}{2\delta^2}\right)\right] - B\right\}C$$

式中 $X=[0,1]$，A、B、C 是常数，网络的期望输出函数为：

$$D = KX + P$$

取 $A=0.5$，$B=0.75$，$C=3$，$K=-5$，$P=12$. 使用 BI 算法，运行结果见下表 3.6，精度达到 99.206%.

表 3.6 使用 BI 算法函数非线性变换的运行结果

输入值	实际输出值	期望输出值	误差
X=2.23924	Y=11.89190	T=12.00000	E=0.10810
X=2.22559	Y=11.87502	T=11.87500	E=0.00002
X=2.18502	Y=11.82476	T=11.75000	E=0.07476
X=2.11860	Y=11.74228	T=11.62500	E=0.11728
X=2.02810	Y=11.62998	T=11.50000	E=0.12998
X=1.91583	Y=11.49186	T=11.37500	E=0.11686
X=1.78462	Y=11.33387	T=11.25000	E=0.08387
X=1.68761	Y=11.16366	T=11.12500	E=0.03866
X=1.47818	Y=10.98988	T=11.00000	E=0.01012
X=1.30977	Y=10.82092	T=10.87500	E=0.05408
X=1.13576	Y=10.66348	T=10.75000	E=0.08652
X=0.95936	Y=10.52170	T=10.62500	E=0.10330
X=0.78350	Y=10.39676	T=10.50000	E=0.10324
X=0.61077	Y=10.28724	T=10.37500	E=0.08776
X=0.44333	Y=10.18966	T=10.25000	E=0.06034
X=0.28294	Y=10.09925	T=10.12500	E=0.02575
X=0.13089	Y=10.01067	T=10.00000	E=0.01067
X=−0.01192	Y=9.91866	T=9.87500	E=0.04366
X=−1.14499	Y=9.81879	T=9.75000	E=0.06879
X=−0.26817	Y=9.70801	T=9.62500	E=0.08301
X=−0.38157	Y=9.58505	T=9.50000	E=0.08505
X=−0.48556	Y=9.45040	T=9.37500	E=0.07540
X=−0.58065	Y=9.30600	T=9.25000	E=0.05600
X=−0.66750	Y=9.15467	T=9.12500	E=0.02967
X=−0.74682	Y=8.99954	T=9.00000	E=0.00046
X=−0.81933	Y=8.84363	T=8.87500	E=0.03137
X=−0.88576	Y=8.68947	T=8.75000	E=0.06053
X=−0.94685	Y=8.53900	T=8.62500	E=0.08600
X=−1.00320	Y=8.39360	T=8.50000	E=0.10640
X=−1.05540	Y=8.25412	T=8.37500	E=0.12088
X=−1.10397	Y=8.12105	T=8.25000	E=0.12895
X=−1.14935	Y=7.99456	T=8.12500	E=0.13044
X=−1.19192	Y=7.87465	T=8.00000	E=0.12535
X=−1.23201	Y=7.76121	T=7.87500	E=0.11379
X=−1.26986	Y=7.65403	T=7.75000	E=0.09597
X=−1.30571	Y=7.55201	T=7.62500	E=0.07209

续表 3.6

输入值	实际输出值	期望输出值	误差
$X=-1.33972$	$Y=7.45761$	$T=7.50000$	$E=0.04239$
$X=-1.37205$	$Y=7.36788$	$T=7.37500$	$E=0.00712$
$X=-1.40281$	$Y=7.28349$	$T=7.25000$	$E=0.03349$
$X=-1.43210$	$Y=7.20418$	$T=7.12500$	$E=0.07918$
$X=-1.46002$	$Y=7.12972$	$T=7.00000$	$E=0.12972$

3.4.4 BP 算法样本特性及参数 α,β 两阶段动态调整

误差逆传播算法,其明显缺点是学习时间过长,甚至难于收敛于全局极小点.虽已有许多旨在提高学习速度的学习算法提出,并在实践中得到应用.但都没有针对这样一个事实:对具有不同特性的输入样本,同一初始权值的 BP 网会表现出不同的学习能力;对同一样本,不同初始权值的 BP 网会表现出不同的学习能力.基于在 BP 算法中加入同一两络对不同样本会表现出不同特性的想法,本节给出了缺席学习算法 BP1 和强化学习算法 BP2.

另外,在考察了 BP 算法的学习曲线与参数 α,β 关系的基础上,提出了两阶段动态调整算法 BP3,通过动态的、自适应的方法,对参数 α,β 进行两阶段调整,以提高学习速度.

针对 XOR 等问题,比较了以上三种算法与原 BP 算法的学习速度,并对它们进行单独和组合测试,取得了较好的结果.

1. 学习算法

在 BP 算法中,逆传播学习通过了一个使全局误差函数最小化过程完成输入到输出的映射.误差函数则定义为所有输入模式上输出层单元期望输出与实际输出的误差平方和:

$$E=\sum_j \frac{(T_j-O_j)^2}{2} \tag{3.4.23}$$

其中 O_j 是输出层单元 j 的实际输出,T_j 是此单元的期望输出.全局误差函数 E 曲面上梯度下降由 E 对权值 W_{ji} 和阈值 θ_j 的微分决定.

对于输出层单元 k,一般化误差 δ_k 可以表示为

$$\delta_k=O_k(T_k-O_k)(1-O_k) \tag{3.4.24}$$

对隐含层单元 j,δ_j 可以表示为

$$\delta_j=H_j(1-H_j)\sum_k \delta_j W_{kj} \tag{3.4.25}$$

输出层单元权值 W_{kj} 与阈值 θ_k 修正如下

$$W_{kj}=W_{kj}+\alpha\delta_k H_j \tag{3.4.26}$$

$$\theta_k=\theta_k+\beta\delta_k \tag{3.4.27}$$

隐含层单元权值 W_{ji} 与阈值 θ_j 修正如下

$$W_{ji}=W_{ji}+\alpha\delta_j I_i \tag{3.4.28}$$

$$\theta_j=\theta_j+\beta\delta_j \tag{3.4.29}$$

在学习过程中,对每个输入样本逐次修正权值和阈值.若有 N 个输入向量,那么一次学习将对权值和阈值修正 N 次,直至误差小于给定的期望值.

2. 针对样本特性的改进算法

在常规 BP 算法中,对学习效果的衡量一般采用一组样本的误差平方和,因此考虑的是整个样本空间的学习情况,并没有考虑到 BP 算法在学习过程中对于每个样本的学习效果.

1)缺席算法 BP1

常规算法在学习过程中对已经学习得非常好的样本仍要进行学习,不但浪费了时间,而且可能对学习得不好的样本产生不良的影响,使得学习次数增加.这样在学习过程中考察每个样本的学习效果是必要的.

基于以上想法,我们引入向量 $\boldsymbol{E}$,称为允许误差.设输入向量样本 $\boldsymbol{X}_k$ 的输出向量为 $\boldsymbol{O}_k$,其对应的教师信号向量为 $\boldsymbol{T}_k$,则当 $\mathrm{ABS}(\boldsymbol{O}_k-\boldsymbol{T}_k)\leqslant\boldsymbol{E}$ 时,网络对样本 $\boldsymbol{X}_k$ 就放弃本次的学习.当网络在某次学习中放弃了所有的样本的学习,则网络对所有的样本的学习都达到要求,算法结束.

原则上,$\boldsymbol{E}$ 中各分量可以不同.我们在实现中简单规定 $\boldsymbol{E}$ 中的各分量都相等,令为 e,则当 $\mathrm{ABS}(\boldsymbol{O}_k-\boldsymbol{T}_k)\leqslant e(k=1,2\cdots,m,\ m$ 为输出层单元个数)时,认为网络对样本 $\boldsymbol{X}_k$ 的学习达到要求.

特别的,由于缺席算法在学习过程中样本有退出学习的情况,所以即使学习次数与常规算法相同时,同样会缩短学习时间.

2)强化算法 BP2

基于 BP 算法对于每个样本的学习能力不同的考虑.其对不同的样本的学习效果如果相差得很大,就可能出现由于对一个或几个样本的学习效果差而影响整个学习进度的情形.因此,可以对学习差的样本在同一次学习中进行强化学习,从而缩短总的学习次数.

同样,设输入向量样本 $\boldsymbol{X}_k$ 的输出向量为 $\boldsymbol{O}_k$,其对应的教师信号向量为 $\boldsymbol{T}_k$,令

$$M_k=\sum_{t=1}^{m}\mathrm{ABS}(O_{kt}-T_{kt})$$

且令

$$M_i=\min_{t=1}^{N}M_t,M_j=\max_{t=1}^{N}M_t$$

N 为输入样本个数.

当 $M_iC\leqslant M_j$(C 为常数,本实验中取 2)时,即当学习效果最好的样本和最差的样本在学习效果上相差一个常数倍时,可以认为样本学习效果相差较大,学习差的样本可能会影响整个学习,则在本次学习结束时,对输入向量样本加强学习一次.

3. 参数 α,β 两阶段调整算法

多层网络对自身参数的选取非常敏感.对同一个问题,网络各个参数稍有不同,学习速度会大不相同.如何正确地选择网络参数值是一个很难解决的问题,如果让网络能自身动态调整参数,就可以无需人工干预,从而达到良好性能.

参数 α,β 对 BP 算法学习速度影响很大.一般来说,α,β 值越大,由误差所得的修正值也越大,学习速度也就越快.但若过大,将引起系统振荡,反而降低性能.最优的选择参数值是很困难的,可以让网络动态调整,已经有这样的算法被提出.但是,以前提出的算法,对参数的调整策略在整个学习过程中是不变的,没有根据学习的具体情况进行相应的变化,影响了算法的性能.通过对大量 BP 算法学习过程的最大误差变化曲线的观察和分析,我们知道,BP 算法的学

习过程大体可分为学习前期和学习后期两个阶段. 据此我们提出参数 α,β 的两阶段动态调整算法:

(1)学习前期,输出信号与教师信号之间的误差一般较大,学习相对较快. 这时,考察最近 P(如 10)次以来的最大误差情况,设为 $E_t(1\leqslant t\leqslant P)$,计算这 P 次的最大误差的平均值 S 及偏差值 E_d:

$$S=\sum_{t=1}^{P}E_t/P,\quad E_d=\sum_{t=1}^{P}\mathrm{ABS}(S-E_t) \tag{3.4.30}$$

若 E_d 小于一定阈值(如平均误差 S 的 1/20),则认为误差已基本不变,这时,将 α,β 分别加大(如 0.1). 若 E_d 大于一定阀值(如平均误差 S). 且不处在误差急剧下降的阶段,则认为出现较大的振荡将 α,β 分别减小(如减 0.02). 否则,认为处于误差急剧下降阶段,这时不对 α,β 进行调整.

(2)学习后期,输出信号与教师信号之间误差变得很小,导致公式(3.4.24)中 δ_k 变得很小,学习异常缓慢. 可以采用如下方法判别,当 $\max\limits_{t=1}^{P}E_t<T$($T$ 为常数,现实中取 0.1)时,即输出信号与教师信号之间误差小于某一阀值 T 时,认为已经进入学习后期. 这时,当误差大小基本不变时,α,β 增大的幅度应当加大(如加 1.0). 为防止 α,β 过大引起振荡,应规定 α,β 的上限(如为 10.0).

4. 性能测试

在 BP 算法中,一个样本的一次学习可以分为两部分:从输入信号推导出输出信号部分和从输出信号开始反向修正网络权值的部分. 前者我们称为检验过程,后者称为修正过程. 常规算法中,一次学习包含一次检验和一次修正. 在我们的算法中,一次学习包含一次检验,而修正次数可以是没有、一次或两次. 这样,在缺席算法中,修正次数小于等于检验次数. 强化算法中,修正次数大于等于检验次数. 在下面的结果测量中,我们分别给出样本的学习次数、修正次数和检验次数.

(1)XOR 学习. XOR 问题是测试 BP 算法中最典型的问题之一. 表 3.7～3.11 给出其样本及各因素对算法的影响. 由表 3.8 可见,不同的隐节点数对各种算法均有影响,BP1、BP2、BP3、ALL 算法对学习速度均有不同程度提高. 几种算法的学习最大误差曲线如图 3.19 所示. 由表 3.9 可见,BP 算法对 α,β 值的依赖性很大,而采用 BP3 算法时,不同 α,β 值对学习成绩影响很小,由表 3.10 可见,学习要求精度高时,BP3 算法和 ALL 算法比其他算法性能要好. 由表 3.11 可见,统计总的修正次数之后,BP1,BP2 算法基本有稳定的加速比,这表明它们具有广泛的应用范围.

表 3.7 XOR 学习样本

样本	输入	输出
1	00	0
2	01	1
3	10	3
4	11	0

表 3.8　学习次数与隐层结点关系

隐节点	BP	BP1	BP2	BP3	BP1&BP2	BP1&BP3	BP2&BP3	ALL
2	881	789	769	283	681	274	224	218
4	887	835	738	273	728	269	233	228
6	577	502	575	178	502	177	175	175
8	416	393	413	165	389	158	163	158

* $\alpha=\beta=0.3$，学习率 $\mu=0.8$，误差 $e=0.05$

表 3.9　学习次数与 α、β 的关系

$\alpha=\beta$	BP	BP1	BP2	BP3	ALL
0.2	>999	>999	>999	284	224
0.4	675	609	582	283	210
0.6	479	434	399	284	207
0.8	395	362	311	289	208

* $\mu=0.8$，误差，$e=0.05$，隐节点数为 2

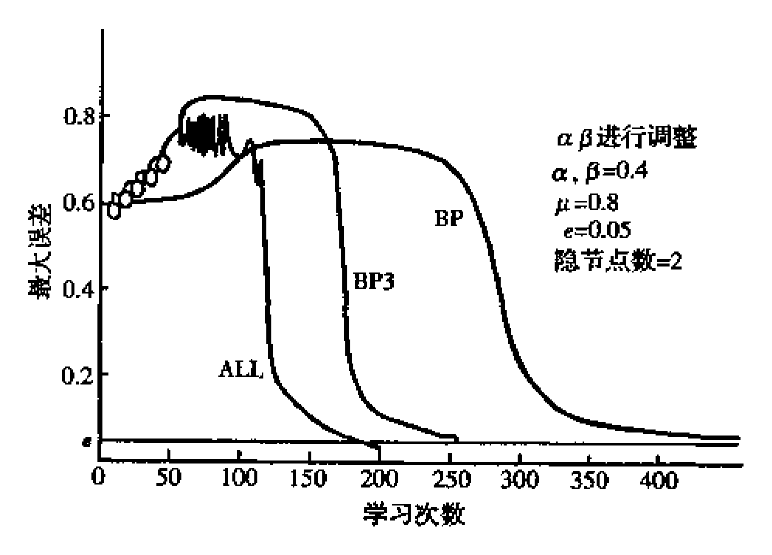


图 3.19　BP，BP3，ALL 误差曲线

表 3.10　学习次数与允许误差关系

误差	BP	BP1	BP2	BP3	ALL
0.2	234	220	149	219	160
0.1	270	256	186	246	150
0.05	395	362	311	289	208
0.02	>999	>999	>999	581	445

* $\mu=0.8$，$\alpha=\beta=0.8$，隐节点数为 2

表 3.11　修正次数与 α、β 关系

α	β	BP	BP1	BP2	BP3	ALL
0.1	0.2	>4000	>3976	>4214	1136	941
0.4	0.3	2984	2448	2711	1100	871
0.6	0.4	2184	1786	1945	1136	866
0.8	0.6	1684	1406	1454	1176	870
0.9	0.7	1552	1308	1306	1212	870

* $\mu=0.8, \epsilon=0.05$，隐节点数为 2

(2)四比特校验问题. 四比特校验问题是对给出的四个 0,1 代码求出其奇偶校验和，也是检验 BP 算法性能的典型问题之一(样本略). 表 3.12 给出了检验次数＋修正次数与 α,β 的关系：

表 3.12　检验次数＋修正次数与 α,β 的关系

α	β	BP	BP1	BP2	BP3	ALL
0.2	0.2	909＋909	810＋533	527＋553	503＋503	315＋206
0.4	0.3	894＋894	769＋392	529＋560	463＋463	334＋210
0.6	0.4	749＋749	659＋311	647＋686	418＋418	389＋231
0.8	0.6	653＋653	612＋278	213＋225	529＋529	257＋167
0.9	0.7	649＋649	285＋155	437＋463	532＋532	158＋129

上述改进在实验中取得了较好的效果，可以看出缺席算法和强化算法的加速比较接近，缺席算法的性能比较稳定，强化算法的学习次数少. 参数动态调整算法性能优于以上二者，特别在学习误差要求精度高的情况下，性能更优. 这些算法的综合，一般比单方法的学习次数少.

综上所述，利用 BP 算法训练神经网络求解一定的问题需要大量的学习次数. 为加快学习过程，可以利用样本特性和参数动态调整的方法. 通过对以上两方面的研究，提出了三种改进算法. 通过对 XOR 等问题学习的测试，验证了上述算法的可行性. 上述改进，可以说，适用于使用 BP 算法求解问题的场合. 由于准确地描述样本特性和参数调整对网络学习的影响是一个复杂的问题，本文仅做一些基本的工作，有待于做更深入的、大量的研究.

§3.5　使用遗传算法(GA)训练前馈型神经网络方法

本节简述了遗传算法的基本原理、操作及算式，并以一个识别部分英文字母的多层前馈型神经网络为例，给出了用遗传算法训练网络、优化网络权值的方法和步骤.

遗传算法(Genetic Algorithm，简称 GA)是模拟自然界生物进化过程的计算模型. 它依据适者生存、优胜劣汰的进化规则，对包含可能解的群体反复进行基于遗传学的操作，不断生成新的群体并使群体不断进化，同时以全局并行搜索方式来搜索优化群体中的最优个体，以求得满足要求的最优解.

GA 比起其他搜索方法，如随机查找、梯度下降、模拟退火等，主要优点是简单、鲁棒性强. GA 实现全局并行搜索，搜索空间大，并且在搜索过程中不断地向可能包含最优解的方向调整搜索空间，以便寻找到最优解或准最优解. 用 GA 解决的问题越复杂，目标越不明确，其优越性

越明显.

GA是由美国学者Holland于1975年首次提出的.近年来,它在组合优化问题求解、机器学习、人工生命等领域已展现了它的应用前景和潜力,在国内外对GA的研究和应用已成为十分热门的研究课题,并已得到了比较广泛的应用.

一、基本概念

GA进行遗传操作的基本对象是个体或染色体(chromosome).每个染色体是一个知识结构,代表所要求解的问题的一个可能解.染色体通常用字符串或位串来表示,若干长度的串称为构成染色体的基因(gene).

群体(Population)由一组(N个)染色体组成,它代表GA搜索的遗传空间.GA对群体中所包含染色体的数量(N)很敏感;从维持群体中个体的多样性、防止陷入局部解的角度来考虑,似乎N越大越好,但是这会明显增加计算量,还可能影响个体竞争.

GA中用适应度函数(fitness function)来评价染色体的优劣.染色体的适应度值越大,相应染色体所代表的解越优,生存的概率越大.选择适应度函数的依据是能否有效地指导搜索空间沿着面向优化参数组合方向,逐渐逼近最佳参数组合,而不会导致搜索不收敛或陷入局部最优,同时要易于计算,但并不像BP算法那样要求函数具有可微性.

二、基本操作

编码(Coding)是将问题解的表示映射成遗传空间解的表示,即用字符串成位串构造染色体,其相反操作为解码.

选择(Selection)操作是根据染色体的适应度,在群体中按一定概率选取可作为父本的染色体,选择的依据是:适应度大的染色体被选中的概率大.

交叉(Crossover)操作是按一定概率随机地选择染色体对,然后对染色体对按一定概率随机地交换基因以形成新的子染色体.

变异(Mutation)操作是按一定概率随机地改变某个染色体的基因值.

三、算法

首先,确定编码形式,定义染色体串的长度、各基因的意义和表示形式及确定映射关系,然后确定适应度函数.

算法的大致步骤如下:

(1)随机产生N个染色体,构成初始群体S.

(2)计算群体S中各染色体的适应度.

(3)依据各染色体的适应度,按一定概率随机地选择K对($1<K<N/2$)染色体,构成子群体S',$S'\subset S$.

(4)随机地将S'中染色体两两配对.

(5)对每对染色体按照某一概率实施交叉操作,形成K对新的子染色体构成的子群体S''.

(6)对S''中的染色体按某一概率实施变异操作.

(7)计算$S+S''$群体中所有染色体的适应度,并淘汰掉适应度小的K对染色体.形成新一代群体S.

(8)若S中适应度最大的染色体满足要求的适应度值成评价标准,成者完成了指定代的搜索,则解码S中适应度最大的染色体,得到问题的求解,否则转向(3).

四、使用训练前馈型神经网络的方法步骤

前向神经网络是无反馈、有教师指导的阶层状网络结构. 对前向网络的学习训练通常采用著名的 BP(Back Propagation)算法. BP 算法虽然能保证网络学习过程最终收敛,但其虽著缺点是学习训练时间长,易收敛于局部极小. 一些改进的 BP 算法虽然不断被提出,但是效果并不明显. 图 3.20 为一识别部分英文字母的前向神经网络,其中输入层节点数为 35,隐层节点数为 16. 输出层节点数为 4. 其中隐层节点 j 的输入加权和为:

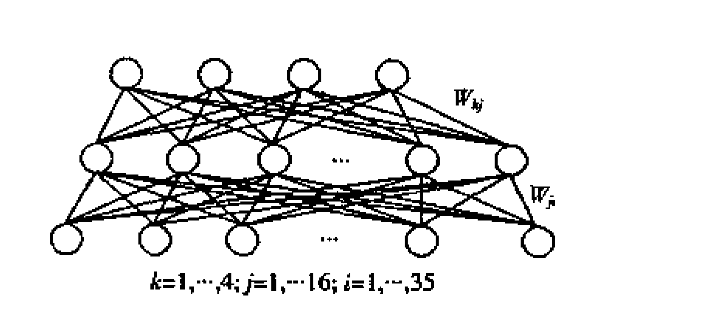
$$u_j(t) = \sum_i w_{ji} \cdot V_i(t) + \theta_j \tag{3.5.1}$$

输出层节点 k 的输入加权和为:

$$u_k(t) = \sum_j w_{kj} \cdot V_k(t) + \theta_k \tag{3.5.2}$$

节点的输出函数用 Sigmoid 函数,即:

$$f(x) = \frac{1}{1 + \exp(-x)} \tag{3.5.3}$$



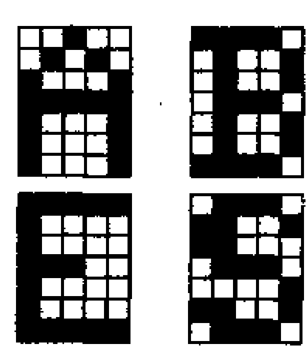


图 3.20 网络结构及训练样本

使用 GA 对网络进行训练优化权值的方法步骤如下:

1. 染色体编码,确定染色体长度和结构

由选定的网络结构可知待训练的网络权值数为:35 * 16+16 * 4+16+4. 每一权值用取值范围为-1.0 至+1.0 之间的数表示,最后将数值的排序固定下来,至此已定义了染色体的结构和长度.

2. 适应度函数的选取

适应度函数选取为:

$$f(i) = 1/E(i) \tag{3.5.4}$$

$$E(i) = \frac{1}{2}\sum_K \sum_P (V_k - T_k)^2 \tag{3.5.5}$$

其中 $i=1,2,\cdots,N$ 为染色体数;$P=1,\cdots,4$ 为输出层节点数;$K=1,2,3,4$ 为学习样本数;T_k 为教师信号.

3. 控制参数的选定

由于 GA 某些控制参数的选定尚无理论上的指导,只能凭经验或实验来确定. 在此,初始群体中的染色体数选取 $N=10$;依据适应度高的染色体被选作父染色体的概率大的原则,选 $K=1$ 对染色体作为父染色体;交叉操作交换染色体中一半数目的基因.

4. 网络学习训练过程

(1)对网络权值编码、排序,构造染色体. 随机产生 10 个染色体,构成初始群体. 每个权值

取－1.0至＋1.0之间的小随机数.

(2)对每个染色体解码得到相应权值，按照公式计算网络的误差函数 $E(i)$ 和染色体的适应度函数 $f(i)$.

(3)按选择概率 Psel 选出一对染色体 C_1 和 C_2.

(4)按交叉概率 Pcross 对 C_1 和 C_2 实施交叉操作，得到子染色体 C_1' 和 C_2'.

(5)分别对 C_1' 和 C_2' 按概率 Pmut 实施变异操作，得到新的子染色体 C_1'' 和 C_2''.

(6)对 C_1'' 和 C_2'' 解码得到相应权值，并按照公式计算出网络的误差函数 E 和染色体的适应度值 f.

(7)在 $N+2\times K$ 个染色体中淘汰 $2K$ 个适应度小的染色体，形成新一代群体.

(8)若满足 $E<0.001$，则选择有最小的误差函数值 $E(i)$ 的染色体，即适应度最大的染色体，将其解码得到网络权值. 至此学习训练结束；否则转(3).

经过大约800代的搜索后染色体的子均适应度趋于稳定. 适应度曲线见图 3.21.

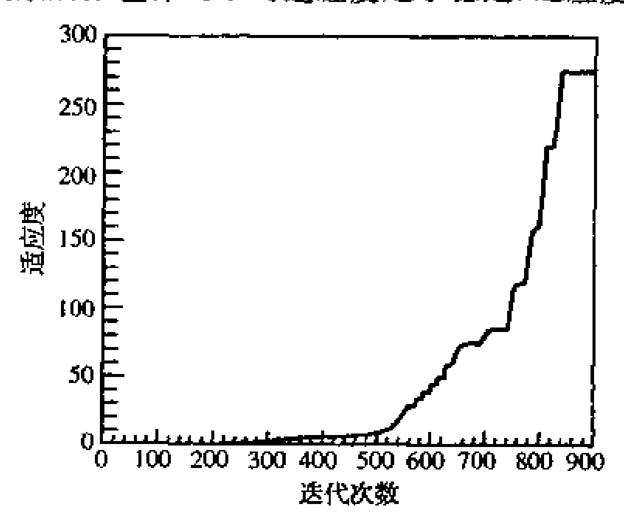


图 3.21 适应度曲线

最优染色体对应的网络结构在训练过程中误差函数 E 的变化趋势见图 3.22(a)所示. 从10个染色体中选取5个染色体，它们对应的网络结构在学习训练过程中误差函数 E 曲线变化趋势见图 3.22(b)所示，可见随着遗传操作的进行，网络的误差函数值逐渐趋于全局最小.

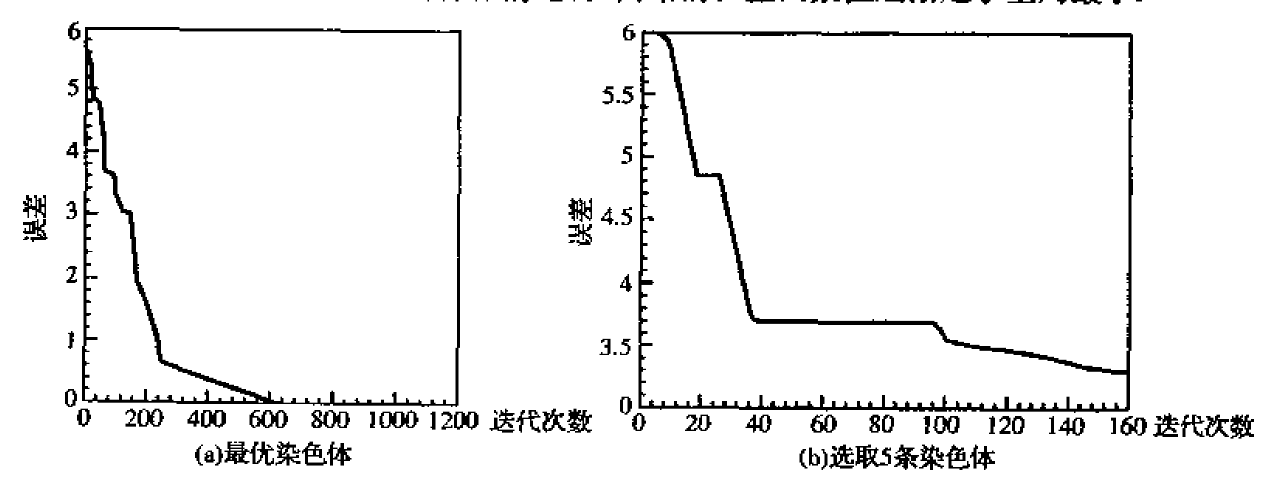


图 3.22 误差函数曲线

用适应度最大的染色体对应的网络参数来识别图 3.20 中的输入样本，识别率达到

100%. 将学习样本加入噪声信号(随机地将样本信号的某些位取反,图 3.23 给出了样本信号附加噪声后的输入信号),作为输入各测量 1000 次,识别率见图 3.24.

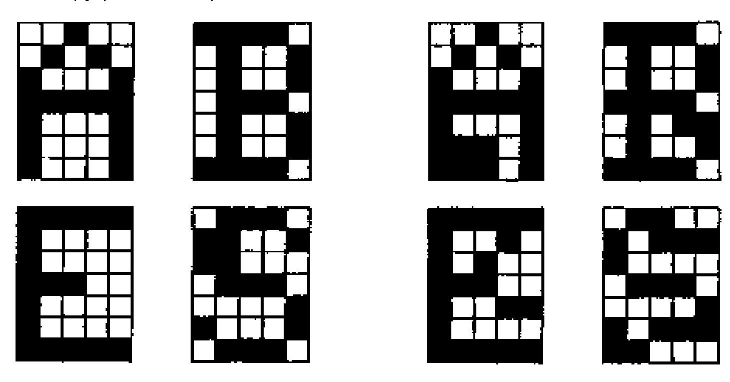


图 3.23 附加噪音的输入信号

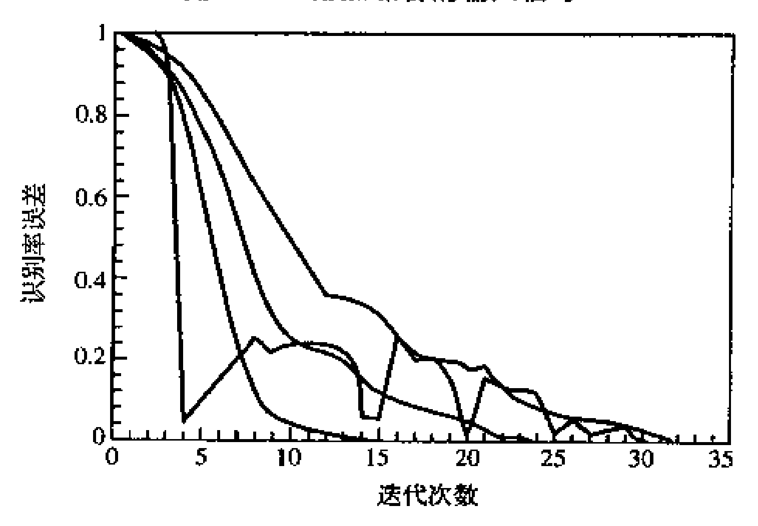


图 3.24 识别率曲线

实验结果表明,用 GA 训练前向神经网络取得了较好的效果,达到了很高的识别率. 实验证明了用遗传算法训练前向神经网络的有效性. 和 BP 算法相比,遗传算法训练时间较长,大约是 BP 算法的 1.5 倍,GA 和 BP 算法相比陷入局部极小的概率低.

总之,遗传算法作为一种全局并行搜索算法,基本框架已经形成,并在各种问题的求解中展现了它的特点和能力,可以预见它将有非常广泛的应用前景. 利用 GA 实现对网络的学习训练的关键是染色体编码方案的确定及适应度函数的选择. GA 存在的主要问题是对于较大规模的神经网络,搜索过程需要较长时间. 另外,GA 所使用的一些控制参数如 P_{cross}, P_{mut} 等目前尚无理论上的指导,还需要进一步探讨与研究.

§3.6 前馈型神经网络结构设计方法

3.6.1 输入层和输出层的设计方法

网络的输入、输出层维数应根据应用的具体要求来设计. 输入层节点数目取决于数据源的维数. 首先要确定正确的数据源,数据源中未经处理的或者虚假的数据将妨碍对网络的正确的训练,所以要剔除那些无效的数据,确定数据源的合适数目. 输入层的节点数可以根据需要求

解的问题和数据表示的方式来确定. 如果输入是电压波形,那么输入单元可以根据电压波形的果样数值和采样点数来决定输入层中单元的维数,也可以用一个单元输入,但输入样本为采样的时间序列. 如果输入为图像,则输入单元可以为图像的象素,也可以是经过处理后的图像特征.

输出层维数应根据使用者的要求来设计,如果网络用作分类器,其类别数为 m,那么一般取 m 个神经元,其训练样本集中的 X_j 属于第 j 类,要求其输出为:

$$T = \begin{bmatrix} \overset{j}{0,0,0,1,0,0,0} \end{bmatrix}^{T}$$

即第 j 个输出为 1,其他输出为 0,因而对一个 n 维输入,$\mathbf{X}\in\mathbf{R}^n$ 进行分类,映射成 $\mathbf{Y}\in\mathbf{R}^m$,满足:

$$Y_j = 1, \qquad X^{P_1} \text{ 属于 } j \text{ 类}$$
$$Y_j = 0, \qquad X^{P_1} \text{ 不属于 } j \text{ 类}$$

此外,输出神经元还可根据类别进行编码,即 m 类的输出只要用 $\log_2 m$ 个输出节点即可.

注意在设计输入和输出层时,应尽可能减小系统的规模,使系统的学习时间和复杂性减小.

3.6.2 隐层数和层内节点数的选择

一、隐层数的选择

隐层数可以采用一个、两个或多个隐层来构造网络结构,隐层数多的网络映射功能增强,但是学习训练时间开销大. 1989 年 Robert Hecht-Nielson 证明了任何在闭区间内一个连续函数都可以用一个隐层的网络来逼近,即一个三层的网络能够实现任意精度的由 n 维到 m 维的非线性映射.

定理 3.4 假定隐层的节点数可以根据需要自由设置,那么用三层 S 状的 I/O 特性的节点,可以以任意精度逼近任何连续函数.

1988 年 Cybenko 指出,当各节点均采用 S 型函数时,一个隐层就足以实现任意判决分类问题,两个隐层则足以表示输入图形的任意输出函数. 目前对二进制分类或判决边界问题,一个隐层就足够了. 但是,如果要求输出是输入的任意连续函数,就要用两个隐层或者采用不同的激活函数. 有时即使是连续输出的情况,用一个隐层也可以满足要求,这取决于问题的性质.

隐层起抽象的作用,即它能从输入提取特征. 增加隐层可增加神经网络的处理能力,但是必将使训练复杂化. 一般来说,开始设定一个隐层,然后按需要再增加隐层数.

增加神经网络处理能力还有另外一些方法,例如在一个隐层的情况下采用多个组. 这种组是模仿大脑的神经元的结构,这些神经元可对视觉输入进行特征鉴别(如鉴别线或者其他特殊的简单图形). 在一层内,多个并行组可用不同类型和不同数目的节点,这种结构可使这些并行组同时提取不同的特征. 大多数应用不要求有多个组的隐层,但是,这种结构对单个隐层的功能有限制的情况,提供了一个解决办法.

二、隐层内节点数的确定

基于 BP 算法的神经网络隐层节点数的选择对网络的性能影响很大,所以,层内节点数需要进行恰当的选择. 隐层单元数的选择是一个非常复杂的问题,目前并没有理论的指导,没有一个好的解析式来表示. 隐层单元的数目与问题的要求、输入、输出单元数都有直接的关系. 一般情况下,隐单元数目太少,可能不能训练出来,或网络不强壮,不能识别以前没有看到过的样

本，容错性差，但是隐单元数太多又使学习时间过长，误差也不一定达到最佳. 一般情况下，仅能靠经验和实验来确定隐单元数.

如果输入输出是二值函数，例如输入为二值图像象素，而输出为 0，1，隐单元数目可以选择和输出层单元数目大致相等.

用作函数逼近的网络，隐层单元数要和逼近函数的精度和函数本身的波动情况有关. 例如：要求逼近精度高，要求逼近的三角函数或多项式的项数要增加，因而隐单元的个数要多，同样如果函数波动越多，也要求增加隐层单元数. 有一种设计是采用逐步增长和逐步修剪法来调整隐单元数目，初始时使用足够多的隐单元，然后在学习过程中逐步修剪掉那些不起作用的隐单元，一直减少到不可收缩为止. 也可以在初始时设定较少的隐单元，学习一定次数后，不成功后再逐步增加隐单元数，一直增加到比较合理的隐单元数为止.

另一种方法是使用遗传算法来确定网络的隐层数和隐层单元数，即把这些待定的数目编码成染色体的基因，然后通过遗传操作来优化网络结构，以确定隐层数和隐单元数.

因为没有很好的解析式表示，可以说隐单元数与问题的要求，输入输出单元的多少都有直接的关系. 根据对隐节点的几何解释，第一隐层的每个节点确定了一个判决而，它把 N 维输入空间（N 为输入向量的分量数）分为两部分. 第二隐层的每个节点又将第一隐层节点形成的多个判决面组合成凸域空间或者判决域. 最后，输出节点又把多个凸域组合成任意形状的判决空间成判决边界. 很明显，隐层的节点有些用来提取输入图形的特征，有些则用来完成某些特殊功能. 试图根据任务来确定隐层节点的数目是很困难的，这是因为网络映射的复杂性和由于许多成功地完成训练过程的不确定性的性质，目前大多数还是以经验为依据.

下面介绍几种方法，可作参考：

(1)1987 年 Hecht-Nielsen 在讨论了具有单隐层的 ANN 的功能之后，指出它可实现输入的任意函数，并提出隐含层节点的数目为 $2N+1$，其中 N 为输入的节点数.

(2)1987 年 Linnmann R P 利用他对多层网络功能的几何解释，提出了对隐含层节点数的估算. 对于一个图形识别问题，假设输出判别边界是任意的形状，那么平均来说，由于每个非凸域的输出边界是靠组合第二隐层的两个凸子域形成的，所以，第二隐层的节点数应为 $M\times2$，这里 M 为输出层的节点数. 在模式分类中，有 $H=\log_2 T$ 的近似关系，其中 H 为隐单元数；T 为输入训练模式数.

(3)1988 年 Kuarycki 根据其实验发现，在高维输入时，第一隐层对第二隐层的最佳节点数比例为 3：1. 例如，由两个隐层的 BP 算法的神经元网络实现图形识别时，设输入节点为 20 和输出节点为 8 时，按 Linnmann 的关系，第二隐层的节点数为 $M\times2=8\times2=16$，根据 Kuarycki 的推论，第一隐层的节点数应为 $3\times(M\times2)=48$ 个.

(4)1990 年 Nelson 和 Illingworth 建议隐含层节点数应为 $4\times N$. 对于图形识别的 ANN，隐含层节点数大为减少，当输入维数较高时，节点数最少可达到 $0.02\times N$. 例如，在贷款评估的网络中，输入节点为 4，那么隐含层节点均取 $4\times4=16$，并在图形识别的例子中，输入节点数为 64，第一隐含层节点数取 64，第二隐含层节点数取 20.

(5)对于医疗诊断，不少人作了研究，认为用具有单隐含层的网络就可得到满意结果. 这种单隐含层网络的输出的判决域通常是凸域. 下面介绍几种选择单隐含层节点数目的估值方法如下：

Lippmann 认为最大隐含层节点的数目为 $M_1(N+1)$.

Kuarycki 认为最大隐含层节点的数目为 $M_1 \times 3$.

Maren A J 等人认为，对小型网络来说，输入节点数大于输出节点数时，最佳隐含层节点数等于输入和输出节点的几何平均值即$(M_1 \times N)^{\frac{1}{2}}$. M_1 为输出层节点数.

(6)下面几个公式也可供选择隐含层单元数参考.

① $K < \sum_{i=1}^{n} C_{n_1}^{i}$ ，K 为样本数，n_1 为隐单元数，n 为输入单元数，如果 $i > n_1$，$C_{n_1}^{i} = 0$.

②$n_1 = \sqrt{n+m+a}$，其中 m 为输出层神经元数，n 为输入层单元数，a 为 1 至 10 间的常数.

③$n_1 = \log_2 n$，n 为输入层神经元数，用于数据压缩的网络. 隐单元数与输入单元数的比为其数据压缩比，常使用此公式.

④还有一种方法就是使隐含单元数可变. 一种是开始放人足够的隐含单元，然后把学习后那些不起作用的隐含单元逐步去掉，一直减少到不可收缩为止.

⑤另一种是在开始放入比较少的隐含单元，学习一定次数后，还不成功就要增加隐含单元个数，一直达到比较合理的隐含单元数为止. 这样做对于用硬件完成 BP 多层网有一定的好处，但是对于结构的选定所花的时间比较长，这属于一种变结构类型.

三、初始权值的选取

由于系统是非线性的，初始值对于学习是否达到全局最小或是否能够收敛关系很大. 一般情况下，希望初始权值在输入累加时使每个神经元的状态接近于零，这样开始时在误差曲面上不落到那些平坦区上. 权值一般取较小的随机数，这样可以保证每个神经元在一开始时都在转移函数变化最大的区域进行.

对于输人样本要进行归一化处理，使得那些比较大的输人值或者太小的输入值不至于使得神经元过于饱和或截止，应能使输人落在神经元转移函数梯度最大的那些区域.

3.6.3 逐次修剪法设计前馈型神经网络

网络结构的选择是一个较复杂的问题，由于问题性质不同，使用的场合和要求也不一样，所谓“最佳”结构没有一个统一的标准，另外，结构空间要比参数大得多，搜索也较困难，一般只能根据实际要求选择一个“满意”的结构即可.

现有的修剪算法很多，大多情况下修剪后的网络还要重新训练. 下面介绍的方法是在修剪中同时调整剩下的权值，以保持整个网络的输出不变，所以不需重新训练，而且所用算法不仅可用于修剪隐节点，也可用于修剪某一连接.

一个前馈网络可用一个无环路的带权有向图 $N=(V,E,\omega)$表示，其中 $V=\{0,1,2\cdots,n\}$是一组 $n+1$ 个节点，$E \subseteq V \times V$ 是一组连接，$\omega: E \to R$ 是一组实值连接权值，ω_{ij}表示节点 i 与 j 间的权，每一节点 $i \in V$ 有一发送域，

$$P_i = \{i \in V:(i,j) \in E\}$$

它是接受节点 i 馈送信号的所有节点集合，还有接收域

$$R_j = \{j \in V:(i,j) \in E\}$$

它是向节点 j 馈送信号的所有节点的集合，以下分别用 p_i 和 r_j 表示 P_i 与 R_j 的大小(节点个数). 节点 V 分为三组输人节点 V_I，隐节点 V_H 以及输出节点 V_O.

图 3.25 是一个典型的前馈网络. 其中 $V_I=\{1,2,3,4\}$，$V_H=\{5,6,7\}$，$V_O=\{8,9,10\}$. 以节点 7 为例，它的接收域 $R_7=\{1,3,5\}$，发送域 $P_7=\{8,10\}$. 网络运行时，输人节点接受外来

信号，非输入节点 i 的净输入为

$$\xi_i = \sum_{j \in R_i} \omega_{ji} y_j \tag{3.6.1}$$

向 P_i 送出的信号为 $y_i = f(\xi_i)$.

算法开始先用一个较大的网络，经 M 个样本训练达到预定的性能，然后逐次删除某些隐单元，同时适当调节剩下的权值以保持网络性能基本不变. 假定单元 $h \in V_H$ 被定为可能删去的节点，则删除后的连接 E_{NEW} 将变为

$$E_{NEW} = E_{OLD} - (\{h\} \times P_h \cup R_h \times \{h\})$$

设 i 是 h 的发送域 P_h 中的一个节点，输入第 μ 个样本($\mu \in \{1,2,\cdots,M\}$)时的净输入是

$$\xi_i^{(\mu)} = \sum \omega_{ji} y_j^{(\mu)} \tag{3.6.2}$$

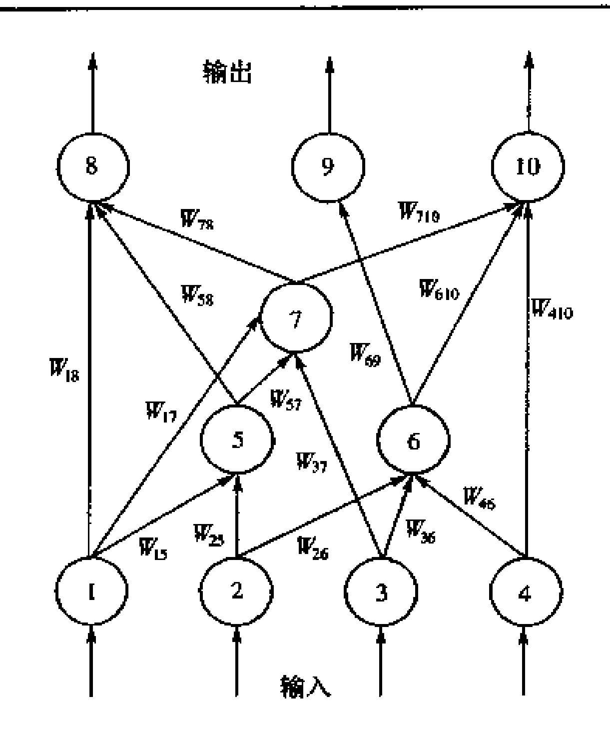


图 3.25　前馈网络的例子

其中 $y_j^{(\mu)}$ 是相应于样本 μ 节点 j 的输出，删掉 h 后节点 i 从 $R_i-(h)$ 得到输入. 为保持网络性能不变，需要调节输入到 i 的其余连接的权值，即所有 $j \in R_i - \{h\}$ 的 ω_{ji} 以使 i 的净输入不变，即希望：

$$\sum_{j \in R_i} \omega_{ji} y_j^{(\mu)} = \sum_{j \in R_i - h} (\omega_{ji} + \delta_{ji}) y_j^{(\mu)} \tag{3.6.3}$$

以上应对所有 $\mu = 1,2,\cdots,M$ 和 $i \in P_h$ 都成立. δ_{ji} 是需要确定的调节量. 上式可写为

$$\sum_{j \in R_i - (h)} \delta_{ji} y_j^{(\mu)} = \omega_{hi} y_h^{(\mu)} \tag{3.6.4}$$

这是 MP_h 个线性方程式，它有 $k_h = \sum_{i \in P_h} (r_i - 1)$ 个未知数$\{\delta_{ji}\}$，其中 k_h 是删掉 h 后送人 h 的发送域的总连接个数.

可把线性方程组(3.6.4)写成矩阵形式，对每一单元 $i \in P_h$，网络输入 M 个样本后，它的 M 个输出是一个 M 维向量 $\mathbf{y}_i$，

$$\mathbf{y}_i = (y_i^{(1)}, y_i^{(2)}, \cdots, y_i^{(M)})^{\mathrm{T}} \tag{3.6.5}$$

用 $\mathbf{Y}_h$ 表示 $M \times (r_i - 1)$ 的矩阵，它的列是 i 的新接收域 $R_i-(h)$ 的输出向量，即

$$\mathbf{Y}_{i,h} = [\overline{\mathbf{y}}_{j1}, \overline{\mathbf{y}}_{j2}, \cdots, \overline{\mathbf{y}}_{jr_i-1}] \tag{3.6.6}$$

所以，要对每一 $i \in P_h$ 通过解方程组来求 $\overline{\delta}_i$，其中

$$\overline{Z}_{i,h} = \omega_{h,i} \overline{\mathbf{y}}_h \tag{3.6.7}$$

把以上方程写在一起，有

$$\boldsymbol{Y}_h \overline{\boldsymbol{\delta}} = \overline{\boldsymbol{Z}}_h \tag{3.6.8}$$

其中：

$$\boldsymbol{Y}_h = \mathrm{diag}(Y_{i1,h}, \boldsymbol{Y}_{i2,h}, \cdots, Y_{iP_h,h}) \tag{3.6.9}$$

$$\overline{\boldsymbol{\delta}} = (\overline{\boldsymbol{\delta}}_{i1}^{\mathrm{T}}, \overline{\boldsymbol{\delta}}_{i2}^{\mathrm{T}}, \cdots, \overline{\boldsymbol{\delta}}_{iP_h}^{\mathrm{T}}) \tag{3.6.10}$$

$$\overline{\boldsymbol{Z}}_h = (\overline{\boldsymbol{Z}}_{i1,h}^{\mathrm{T}}, \overline{\boldsymbol{Z}}_{i2,h}^{\mathrm{T}}, \cdots, \overline{\boldsymbol{Z}}_{P_h,h}^{\mathrm{T}}) \tag{3.6.11}$$

且 $i_k (k=1,\cdots,P_h)$ 遍历 P_h.

为解(3.6.8)式，可用最小二乘方法，即

$$\min \| \overline{Z}_h - Y_h \overline{\delta} \|_2 \tag{3.6.12}$$

每删去一个节点都要解(3.6.8)式，所以要有一个非常有效的算法. 用一种共轭梯度法可以解决这一问题.

还有一个问题是如何选择应删去的单元. 选出的根据是选择对网络影响小的单元 h，具体标准为

$$h = \arg \min_{h \in V_H} \sum_{i \in P_H} \omega_{h,i}^2 \| \overline{y}_h \|_2^2 \tag{3.6.13}$$

于是整个删除的步骤可归纳如下：

①$k=0$；

②按(3.6.13)式确定网络 $N^{(k)}$ 中应删去的单元；

③用共轭梯度法解决问题(3.6.12)以求 $\overline{\delta}$ 值；

④构成新网络 $N^{(k+1)}=(V^{(k+1)},E^{(k+1)},\omega^{(k+1)})$

其中

$$V^{(k+1)} = V^{(k)} - \{h\} \tag{3.6.14}$$

$$E^{(k+1)} = E^{(k+1)} - (\{h\} \times P_h^{(k)} \cup R_h^{(k)} \times \{h\}) \tag{3.6.15}$$

$$\mu^{(k+1)} = \begin{cases} \mu_{ji}^{(k)}, i \in P_h \\ \mu_{ji}^{(k)} + \overline{\delta}_{ji}, i \in P_h \end{cases} \tag{3.6.16}$$

⑤$k=k+1$.

⑥重复②～⑤，直到 $N^{(k)}$ 性能明显下降时停止.

§3.7 基于BP算法的前馈型神经网络在识别问题中的应用

前馈型神经网络在现实中的应用很广泛，诸如信号识别、手写文字识别、图表参数识别等. 基于BP算法的前馈型神经网络可更好地完成这些应用功能. 下面列举几个现在较为成熟的基于BP算法的前馈型神经网络应用的例子.

3.7.1 味觉信号的学习和识别

一、味觉信号的特征抽取

众所周知，酸、甜、苦、香、咸被称为基本味. 在对五种基本味觉信号的模式识别中，首先通过味觉传感器对味觉信号进行数据采集，对每种味觉信号采样的数据点相当于一个 200×41 阶的实数矩阵，而不能直接用味觉传感器的测量值作为输入样本. 此时，需要通过适当的变换方法，对味觉采样信号进行特征抽取和特征选择，以降低样本的空间维数，即需要通过对原始测量值进行压缩、舍弃，提取能表征五种味觉信号的明显特征.

图 3.26 是对酸味和甜味的测量数据绘制的三维图形. 不难看出，对五味的识别也可以归结为对三维图形的模式识别.

1. 特征抽取方法

从广义上说特征抽取，就是一种变换. 若 $\boldsymbol{Y}$ 是测量空间，$\boldsymbol{X}$ 是特征空间，则特征抽取 A 即变换为：$A:\boldsymbol{Y}\to\boldsymbol{X}$.

使用分割取块的变换方法，先将采样矩阵(图形)分割成规则的块，再计算每块内采样值的

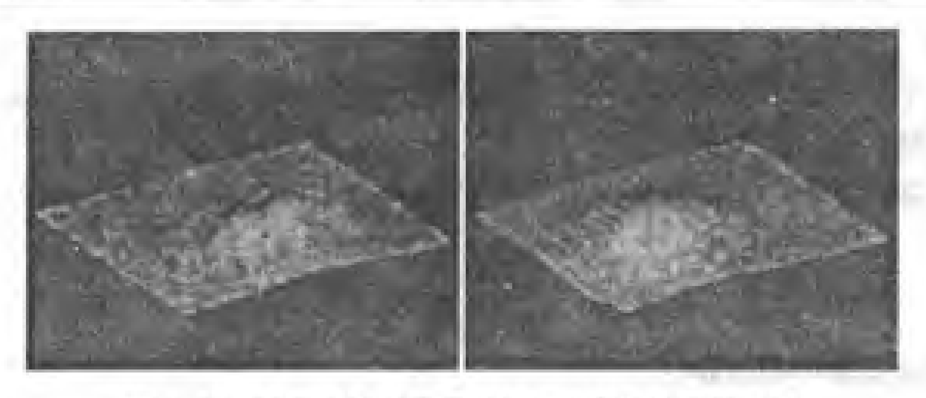


图 3.26　利用对酸味和甜味的采样数据绘制的图形

累加和(该块的体积)作为该块的表征，以达到降低输入样本维数的目的．从采样矩阵的大小及保证分割后每块内所含的采样点数大体相近考虑，选定五种分割方法(见图 3.27)，至于哪种分割方法好，有待通过实验加以验证．如果按图 3.27(a)的分割方法，任取一子块 n_i，假定其物理位置是从 a 行起 b 行止，c 列起 d 列止，并把该位置映射到五种味觉信号相应位置上的五个子块记为 $n_{ij}(j=1,2,\cdots,5)$，则味觉信号 j 在子块 n_i 处的累加和为：

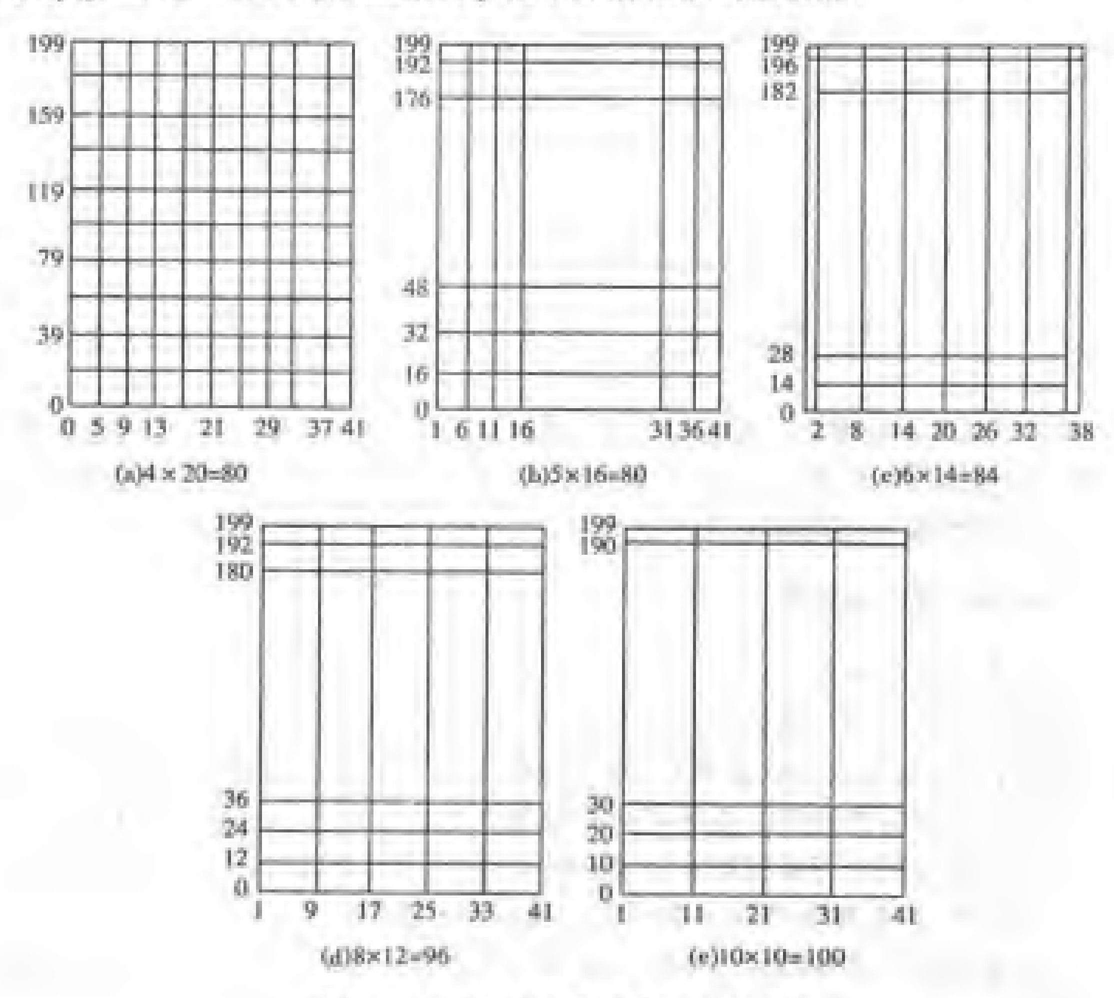


图 3.27　实验用的 5 种分割取块的分法

$$V_{ij} = \sum_{x=a}^{b}\sum_{y=c}^{d} f(X,Y) \tag{3.7.1}$$

计算出全部 V_{ij}，于是味觉信号 j 可以用一个 10×10 的矩阵来替代，从而大大降低了输入样本的分量数．

为了消除味觉信号测量过程中采样值受味觉液体浓度因素的影响，可用相对值替代绝对值加以解决，即先做和：$V_j = \sum_i V_{ij} (j = 1,\cdots,5)$，$j$ 为块数；再求相对值：$V'_{ij} = V_{ij}/V_j$．

这样就可以得到一个不受液体浓度影响，以相对值为元素的矩阵作为网络的学习样本，从而达到了仅对五味识别的目的.

2. 特征选择

从图 3.26 可以看出，图形的某些部分并没有明显的分类特征，因而可以删去那些没有明显特征的块，达到进一步降低空间维数的目的.

为了选择有各自明显特征的块，任意子块 n_i 映射到五种味觉信号相应位置上的 5 个子块记为 $n_{ij} (j=1,\cdots,5)$，它们之间的区别程度用特征值 T_i 来表示，即 T_i 值大，则说明五种味觉信号在于块 n_i 处各自特征明显. 使用下面两种方法求 T_i 值.

(1)在子块 n_i 处，若各子块的体积为 $V_{ij} (j=1,\cdots,5)$，

$$V_i = \sum_{j=1}^{5} V_{ij}, V'_{ij} = V_{ij}/V_i, j = 1,\cdots,5 \tag{3.7.2}$$

则 T_i 值为：

$$T_i = \min(|V'_{ij} - V'_{ip}|), k \neq p \quad p,k = 1,\cdots,5 \tag{3.7.3}$$

(2)倘若各子块在 n_i 处体积都相等，则其 V'_{ij} 值均为 0.2；反之，各子块相互间差异大，则 V_{ij} 值都远离均值 0.2，显然可用下式求 T_i 值：

$$T_i = \sum_{j=1}^{5} |V'_{ij} - 0.2| \tag{3.7.4}$$

之后，按所求的各子块的 T_i 值大小，将子块排序，选择时可保留大的，舍弃小的. 排序时，可以按不同的分割方法各自排序，也可以混在一起排序，当然这时可能出现所选的子块部分重叠. 至于选择多少块，由实验来确定.

3. 实验与结果

对味觉信号的特征抽取和选择是在 SUN 工作站上用 C 语言实现的. 由程序实现对各子块 T_i 值的计算和排序，而后的输出文件记载着按 T_i 值排序的各子块的位置信息，以便在学习训练和识别时，进行子块的选取.

对五味信号的识别是基于 BP 神经网络的方法，即先将压缩后的五个学习样本作为网络的输入，经不断调整网络的权值矩阵，待学习结束后，再对未知味觉信号进行识别.

特征抽取的效果可以从学习次数、时间和识别精度来评价. 首先，按图 3.27 的分割方法，对每种分割各自排序，各取 T_i 值大的前 25 块作为输入样本，测量其学习次数和错判次数(包括误判和漏判). 由测试结果图 3.28 可知，分割方法(a)为最佳.

接着进一步考察分割(a)，改变输入样本的分量数，即改变选择块数量测定学习次数，见图 3.29，可见按 T_i 值选取前 40 块时，学习次数最少. 图 3.30 是对五种分割的块按 T_i 值混在一起排序，随着所选择的块数的增加，测量其学习次数和时间所得到的曲线. 可见，当按 T_i 值选择前 15 块时，无论从学习次数和时间来看都是最小的. 图 3.31 是混合排序和按(a)分割时排序之间的对比，由(b)可知随着所选择块数的增加，学习时间有增长的趋势，这无疑是因为增加了网络输入单元数，因而加大了计算量的缘故.

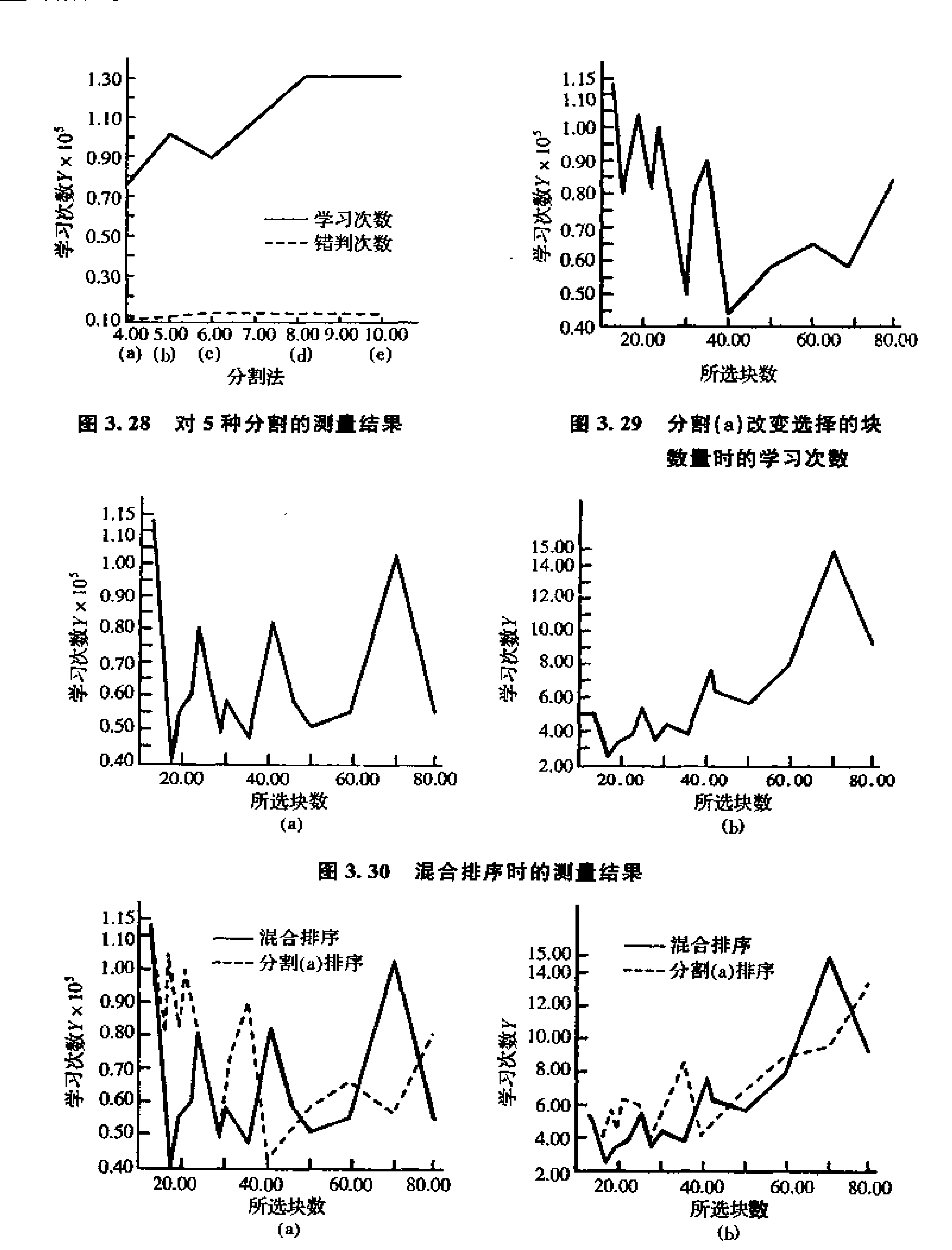


图 3.28　对 5 种分割的测量结果

图 3.29　分割(a)改变选择的块数量时的学习次数

图 3.30　混合排序时的测量结果

图 3.31　两种排序的对比

模式的特征抽取和选择的效果可以用对味觉信号的识别精度，即识别率来评价．可以用测量错判次数的方法模拟实现，即当学习结束后，再在学习样本信号上附加随机噪声信号．若学习信号为 A，C 为 -1 到 $+1$ 间的随机数，则叠加噪声信号后的测量值 $A'=A+0.5CA$．以 A' 为未知信号输入，各测量 1000 次，测量错判次数，结果见图 3.32．由图可知，当按(a)分割排序，

取前 40 块的计算值作为网络的输入样本时，误识率为 4% 左右；当混合排序取前 15 块时，误识率为 12.5% 左右. 两者都能达到设计目标. 另外，还可以看出，随着所选块数的增加，误识率将逐渐降低，显然这是因为所选取的块数越多越逼近原图像所致.

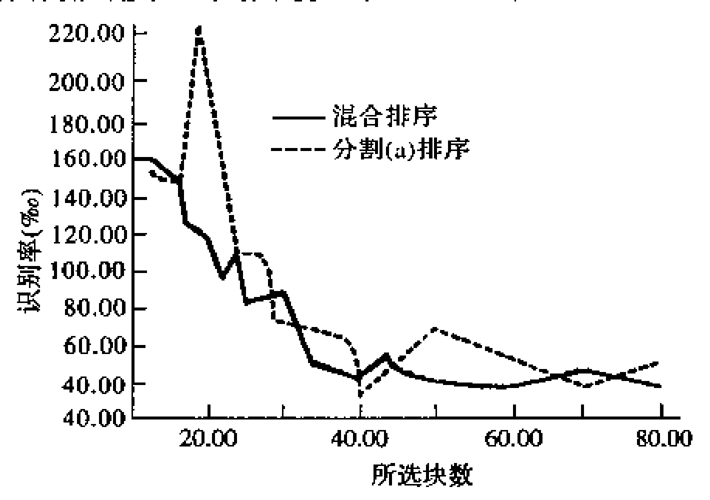


图 3.32　两种排序的误识率

总之，对味觉信号的模式识别，使用了较为简单的分割取块方法进行特征抽取和选择，取得了较满意的效果. 这种方法，无疑也能应用于对一般三维图形进行识别的场合.

二、神经网络味觉信号的学习和识别

1. 样本的选取

由味觉传感器分别对五种味道液体进行数据采样，每种液体一次采样的数据点为 8200 个，如果原封不动地作为神经网络的输入，则使输入神经元数目过多，经模式抽出规范为表 3.19 的形式，其中每列数据对应一种味觉信号的输入模式，将其作为网络的输入信号，即学习样本.

表 3.13　神经网络输入规范模式

模式号	酸味	甜味	苦味	香味	咸味
0	0.196909	0.255833	0.257407	0.144757	0.182608
1	0.200618	0.252565	0.255801	0.151254	0.185557
2	0.488789	0.302145	0.685884	0.738045	1.016772
3	0.230964	0.293088	0.281402	0.173610	0.204479
	⋮	⋮	⋮	⋮	⋮
22	0.222085	0.268812	0.263620	0.164859	0.186757
23	0.301208	0.367970	0.349208	0.243108	0.271468
24	0.616016	0.610636	0.367740	0.566409	0.321578

教师信号如表 3.14 所示，其中每列对应一种味觉信号. 将输入信号和教师信号一同输入给网络，网络在学习过程中将不断地调整权值 W，当输出信号和教师信号间的误差之和 E 小于初始设定值时，学习过程即告结束. 此时将最终调整好的权值 W 存入文件中以待识别时使用. 学习过程如图 3.33 所示.

表 3.14 教师信号

酸味	甜味	苦味	香味	咸味
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

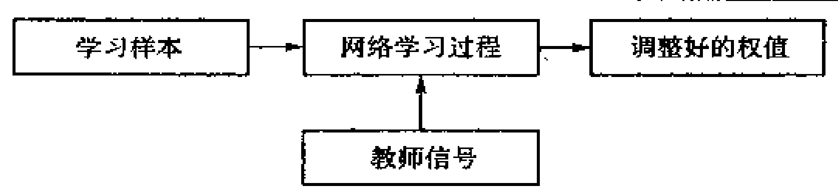


图 3.33 学习过程

2. 味觉信号的学习

三层结构的神经网络如图 3.34 所示，其中输入层神经元数为 25 个(以便和输入模式相匹配)，中间层神经元数由实验确定，输出层神经元数为 5 个. 设输入层神经元输出为 $I_i(i=0,1,\cdots,24)$，各 I_i 分别和输入层至中间层某个神经元 j 间的权值 W_{ji} 乘积之和为神经元 j 的输入信号，该信号与神经元 j 的阈值 θ_j 之和为 U_j，以 U_j 为自变量的 Sigmoid 函数值 $f(U_j)$，即为神经元 j 的输出 H_j，求出中间层全部 H_j. 同样，各 H_j 再分别和输出层某个神经元 $k(k=0,1,\cdots,4)$ 和中间层神经元间的权值 V_{kj} 的乘积之和作为 k 的输入信号，再与 k 的阈值 γ_k 之和作为函数 f 的自变量 S_k，函数值 O_k 即为神经元 k 的输出. Sigmoid 函数为：

$$f(x) = 1/(1+\exp(-x/U_0)) \tag{3.7.5}$$

U_0 为决定函数斜率的常量，即学习率.

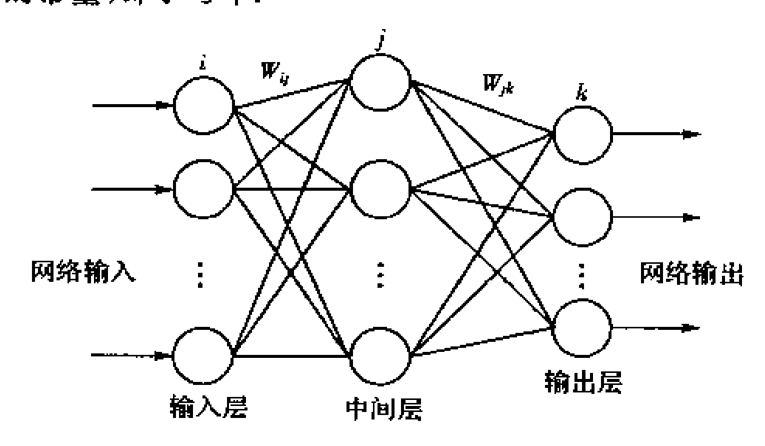


图 3.34 网络结构

具体的学习算法按下面顺序进行：

(1)网络参数的初始化. 决定网络状态的权值 W_{ji}，V_{kj}，阈值 θ_j，γ_k 用小随机数初始化；

(2)按序取 1 列模式作为网络的输入信号；

(3)中间层各神经元输出的计算式为：

$$U_j = \sum W_{ji} I_i + \theta_j \tag{3.7.6}$$

$$H_j = f(U_j) \tag{3.7.7}$$

(4)输出层各神经元输出的计算式为：

$$S_k = \sum V_{kj}H_j + \gamma_k, \qquad O_k = f(S_k) \tag{3.7.8}$$

以上为正向传播过程，下面给出误差逆传播过程.

(5)计算输出层神经元的误差. 用教师信号 T_k 和输出层神经元的输出 O_k 间的差值来计算和神经元 k 相关联的权值 V_{kj} 和阈值 γ_k 的误差：

$$\delta_k = (O_k - T_k)O_k(1 - O_k)$$

(6)计算中间层神经元的误差. 用 δ_k 和 V_{kj} 及中间层的输出 H_j 来计算和中间层神经元 j 相关联的权值 W_{ji} 和阈值 θ_j 的误差：

$$\sigma_j = \sum \delta_k V_k H_j(1 - H_j) \tag{3.7.9}$$

(7)输出层神经元的阈值 γ_k 和权值 V_{kj} 的修正. 用误差 σ_j 和中间层神经元的输出 H_j 及常数 α 之积来修正 V_{kj}，并用误差 δ_k 和常数 β 之积来修正 θ_j：

$$V_{kj} = W_{ji} + \alpha\,\delta_k H_i, \quad \gamma_k = \gamma_k + \beta\,\delta_k \tag{3.7.10}$$

(8)中间层神经元的阈值 θ_j 和权值 W_{ji} 的修正. 用误差 σ_j 和输入层神经元的输出 I_i 及常数 α 的积来修正 W_{ji}，并用 σ_j 和常数 β 之积来修正 θ_j：

$$W_{ji} = W_{ji} + \alpha\,\sigma_j I_i, \quad \theta_j = \theta_j + \beta\sigma_j \tag{3.7.11}$$

(9)取下一个输入模式作为输入信号，如果输入模式轮序一周，则计算误差和 E：

$$E = \sum |O_k - T_k| \tag{3.7.12}$$

如果 E 值小于指定的误差范围，则学习终止；否则更新学习次数，返回(2). 如果仍有输入模式待输入则返回(3).

上式中的 U_0, α, β 为事先指定好的常数，其选择将决定学习过程的快慢. 以上学习过程在 SUN 工作站上用语言来实现. 图 3.35 为改变中间层神经元数对学习次数和学习时间的测量结果，可见选择中间层神经元数为 30 时将取得最佳效果.

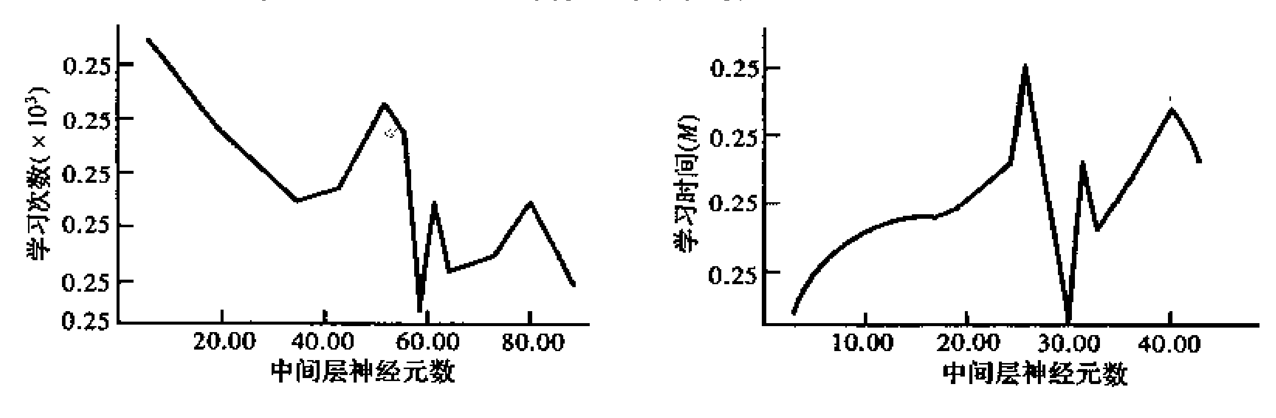


图 3.35　中间层神经元数和学习次数及学习时间的关系

3. 味觉信号的识别

使用神经网络对味觉信号的识别过程如图 3.36 所示. 当待测信号为五种液体味觉信号时，其输出结果如表 3.15 所示，当待测信号为未知液味觉信号时，其测量结果为：0.963892，0.011745，0.010409，0.020601，0.010000. 识别结果为：酸味.

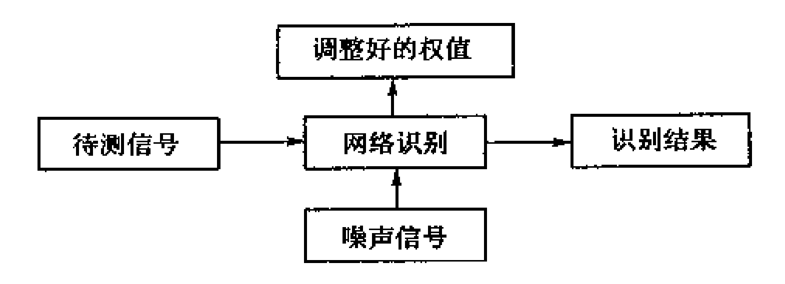


图 3.36 味觉信号识别过程

表 3.15 测量结果

待测信号	输出结果				
酸味	0.952079	0.022198	0.021339	0.020423	0.010000
甜味	0.028328	0.981666	0.010000	0.010000	0.010000
苦味	0.010678	0.010000	0.990000	0.010000	0.010000
香味	0.010000	0.010000	0.010000	0.987471	0.038766
咸味	0.010000	0.010000	0.014723	0.010000	0.990000

上面的结果是使用标准液体信号，即学习过程中所使用的输入信号，但是在实际测量过程中输入信号往往偏离标准信号. 为考证神经网络对味觉信号的识别精度，可在标准信号上附加随机噪声信号作为输入信号. 设 A 为标准信号值，B/A 为信噪比，C 为 −1～+1 间的随机数，那么附加噪声信号后的输入信号值 A 为：$A'=A+C(B/A)$，下面给出当 B/A 取 0.2 时，对五种液体信号各测量 1000 次时的测量结果. 仅有两种液体信号各失败二次. 识别结果为：

1. →(1000,0,0,0,0)

2. →(0,1000,0,0,0)

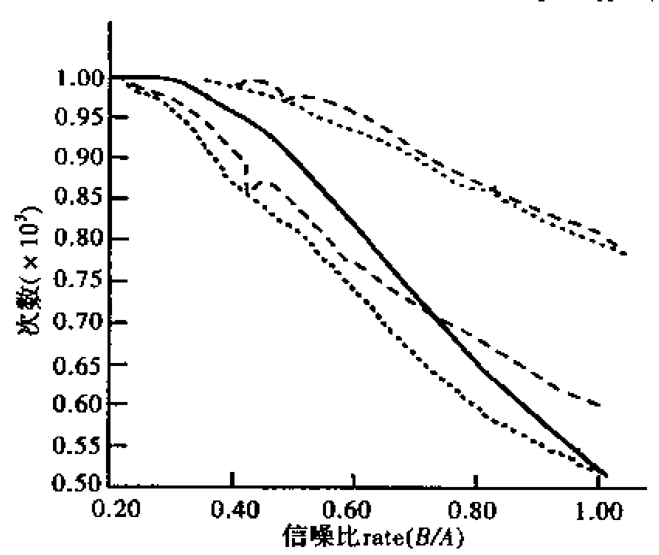
3. →(0,0,998,0,0)

4. →(0,0,0,998,0)

5. →(0,0,0,0,1000)

测量次数=1000，信噪比(B/A)=0.200000

图 3.37 为对五种味觉信号，B/A 取值为 0.0～1.0，间隔为 0.01 时，各测量 1000 次时的测量结果. 由图 3.37 可以看出当信噪比 B/A 为 0.2 时，误测 j 次数将逐渐增多.

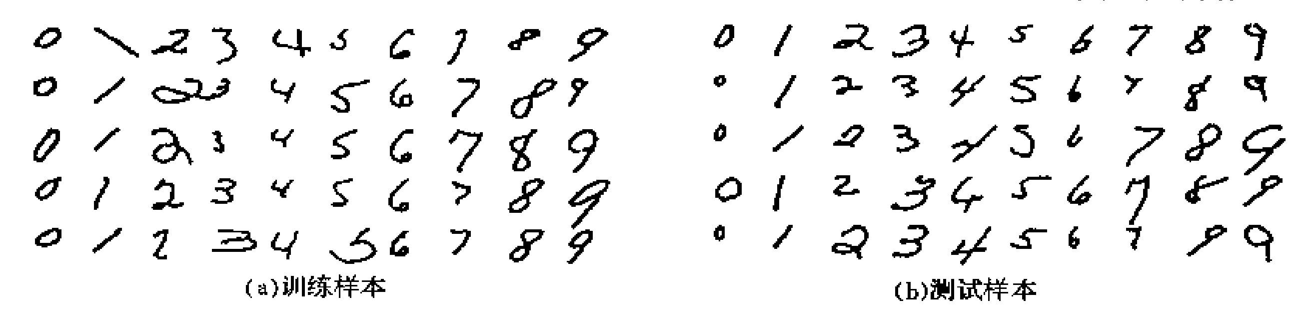


面 3.37 附加噪声信号的识别

综上所述，将神经网络应用于对味觉信号的学习和识别上取得了较为满意的效果，特别是当信噪比小于 0.2 时识别率几乎接近百分之百. 由于神经网络具有较强的自学习和联想记忆能力，与传统的模式识别方法相比，无论是从模式的特征抽取、表示、推理，还是从识别率等方面，都有明显优势和发展潜力. 但是本研究工作仅对五种标准液体信号进行学习和识别，至于混合液体的味觉识别和模糊定量识别，无论从传感器的研制来看，还是从神经网络的实现来看，估计将是十分复杂和困难的事，有待于进一步的研究.

3.7.2 手写体数字识别

手写体数字识别在邮政编码识别、银行业务等方面有重要应用. 由于字体变化大，对识别率要求高，所以有较大困难. 神经网络具有学习能力和快速并行实现的特点，可以用于解决这个问题. 下面介绍的方法是以美国邮局从不同地区的大量信封上搜集到的 6000 个数字作为样本集(图 3.38)，其中 4000 个作为训练集，2000 个作为测试集，以下具体介绍其主要内容.



(a)训练样本　(b)测试样本

图 3.38　手写体数字样本集

1. 特征提取

数字经扫描输入，二值化并统一规格化为 16×16 的图像. 不管是印刷还是手写数字，都是一些线划，即二维平面上的一维结构，可利用局部有向线段作为特征，为检测此特征可利用 Kirsch 模板(或称 Kirsch 算子)，它是一种简单面直观的阶跃边线检测算子. 对数字图像中每一象素(i,j)，考察它 8 个邻点的灰度变化，以其中三个相邻点的加权和减去剩下五个邻点的加权和，令三个邻点环绕(i,j)不断位移，取其差值. 大于一定阈值者说明该方向有边缘存在，可表示为：

$$K(i,j) = \max\left\{1, \max_{k=0}^{7}[\,|\,5S_K - 3T_K\,|\,]\right\}$$

其中

$$S_k = A_k + A_{k+1} + A_{k+2}$$
$$T_k = A_{k+3} + A_{k+4} + A_{k+5} + A_{k+6} + A_{k+7}$$

式中 A_k 为该点灰度值，A 的下标按模 8 计算(图 3.39). 此处只用了四个方向的特征，分别是水平(H)、垂直(V)、右对角线(R)和左对角线(L)：

$$K(i,j)_H = \max(|\,5S_0 - 3T_0\,|, |\,5S_4 - 3T_4\,|)$$
$$K(i,j)_V = \max(|\,5S_2 - 3T_2\,|, |\,5S_6 - 3T_6\,|)$$
$$K(i,j)_R = \max(|\,5S_1 - 3T_1\,|, |\,5S_5 - 3T_5\,|)$$
$$K(i,j)_L = \max(|\,5S_3 - 3T_3\,|, |\,5S_7 - 3T_7\,|)$$

进一步，再把 16×16 上的方向特征压缩为 4×4. 此外再加上总体的轮廓特征(由 15 个付氏描述子与一些简单的内轮廓的拓扑特征组成)，这样共有 5 个 4×4 的特征图(图 3.40).

A_0	A_1	A_2
A_7	(i, j)	A_3
A_6	A_5	A_4

(A)

5	5	5
-3	0	-3
-3	-3	-3

-3	-3	-3
-3	0	-3
5	5	5

(a)

-3	5	5
-3	0	5
-3	-3	5

-3	-3	-3
5	0	-3
5	5	-3

(b)

-3	-3	5
-3	5	5
-3	-3	5

5	-3	-3
5	2	-3
5	-3	-3

(c)

-3	-3	-3
-3	0	5
-3	5	5

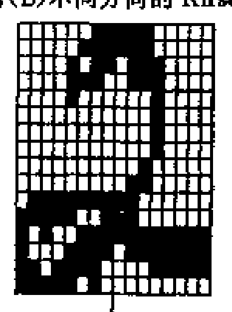
5	5	-3
5	0	-3
-3	-3	-3

(d)

(B)

图 3.39　Kirsch 算子

(A)象元(i,j)的 8 邻域；(B)不同方向的 Kirsch 算子



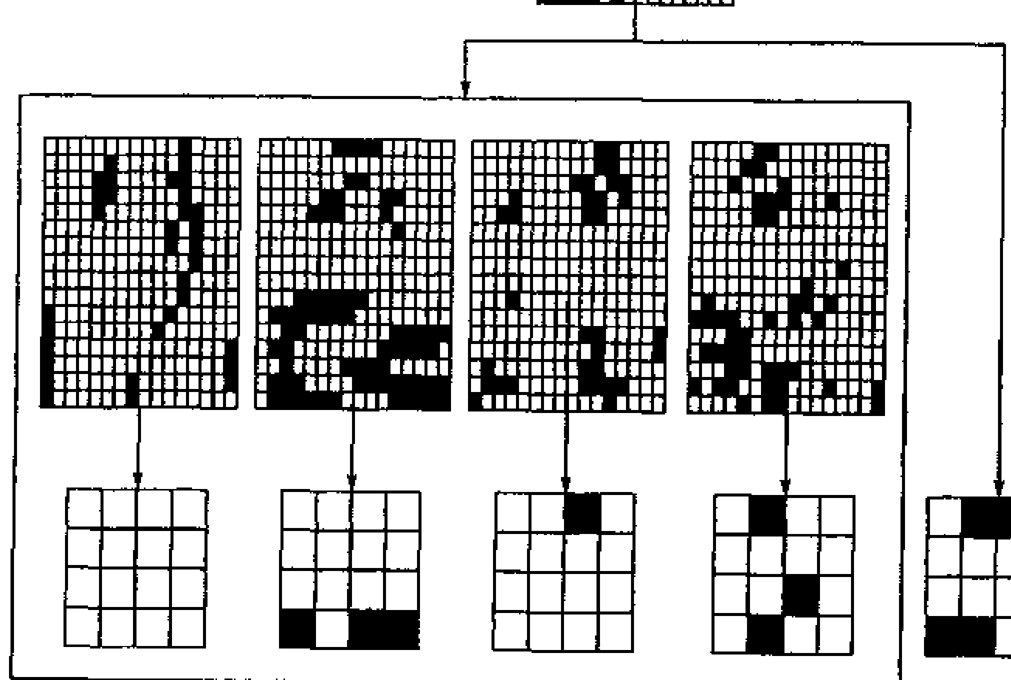


图 3.40　特征抽取

2. 神经网络分类器

基本网络是带一个隐层的前向网络，存储单元及输出单元用 Sigmoid 函数，输出一共分为 10 类. 对于手写体数字识别，其特征数和类别数都较大，且各特征又不是互相独立的，如果都用一个网络来分类，则网络非常复杂，难以训练且推广效果不好，为此采用多网络组合的方法（图 3.41），每个子网络只利用一部分特征，各网络的输出经适当组合方式作为识别结果，组合方式可用加权平均法，少数服从多数的竞选法等等，此处是利用一种模糊集成（Fuzzy internal）方法.

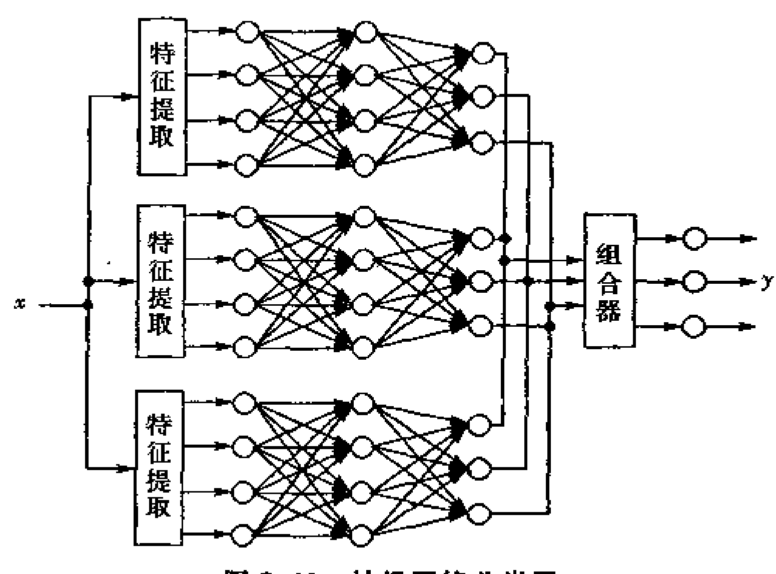


图 3.41 神经网络分类器

3.7.3 在包装件缓冲垫层非线性识别中的应用

缓冲包装设计的基本步骤之一是选择适当的包装件缓冲垫层. 由于缓冲材料几乎都是非线性的，且大多呈强非线性，因此无论从理论研究还是从工程应用的角度，确定包装件缓冲垫层材料的非线性，对于合理地设计包装件，提高产品包装的防震、缓冲能力都是十分重要的. 本节使用一种将遗传算法与 BP 算法相结合的混合训练方法，并将其应用于包装件缓冲垫层非线性特性识别问题中，使得结构化神经网络方法更适于实际应用，为有效地识别缓冲材料的非线性特性提供了一种可行的途径.

一、结构化神经网络模型

在振动工程领域中，一个常见而又非常重要的问题是通过实测数据识别动力系统的非线性特性. 由于人工神经网络模型具备传统计算模型所没有的固有特性，已使得它成为动力学模型识别的一个理想选择. 常见的神经网络方法识别系统非线性的基本原理是相同的，即将系统视作"黑匣子"，用输入、输出样本对训练. 它不考虑系统内部结构，而是将线性与非线性恢复力作为一个整体来识别，得到的是线性与非线性两者的综合动态特性，但是在据要特别了解非线性本身的属性时，这种方法则不足以胜任. 有一种识别振动系统非线性特性的结构化神经网络方法，将系统分为线性与非线性两部分，经学习得到的神经网络可以单独识别出系统非线性模型，而不是线性与非线性综合在一起的模型. 将遗传算法与结构化神经网络方法相结合，并将其用于识别包装件缓冲垫层非线性特性识别问题中，可提高网络训练速度，减少对于训练参数的人为干预，增大搜索空间，容易得到全局最优解，从而使得结构化神经网络方法更适于实际应用.

首先将包装件简化为单自由度模型，其动力学方程可以写为：

$$m\ddot{x} + c\dot{x} + kx + p(x) = f(t) \tag{3.7.13}$$

式中 m 为包装容器与内装物质量，c，k 分别为缓冲包装材料的线性阻尼与线性刚度系数，$f(t)$ 为包装容器在实验条件下承受的激振力，$p(x)$ 为关于位移的非线性项，它是刻划缓冲包装材料的非线性特征的函数.

令 $\dot{x}=y$，并利用后退欧拉公式，可将(3.7.11)式化为：

$$\begin{cases} x(n+1) = x(n) + \Delta t y(n) \\ y^*(n) = y(n) + \Delta t[f(n) - cy(n) - kx(n)]/m \\ y(n+1) = y^*(n+1) - \Delta t p(x(n))/m \end{cases} \tag{3.7.14}$$

假定缓冲包装材料的线性特性可由测量或数值分析方法获得，为识别系统的非线性特性，可将系统分为线性和非线性两部分. 线性网络可以通过两层网络实现，其连接权值可以由(3.7.14)式前两式直接获得；非线性网络用来识别非线性特性，通过三层网络实现，其连接权值由训练获得. 网络训练阶段的目的是通过调节网络内部连接权值使网络运算结果与系统实际输出之间的误差达最小. 采用 BP 算法对网络进行训练. 由于 BP 算法在训练多层网络时收敛速度较慢，且受到参数选取人为因素过多的限制，因此果用将遗传算法与 BP 算法相结合的对于结构化神经网络进行训练的混合方法，用以识别缓冲材料的非线性特性.

二、GA 与 BP 相结合的混合训练方法

遗传算法和人工神经网络是将生物学原理应用于科学研究的典型方法. 由于它们极强的解决问题的能力，近年来引起了众多科研人员和工程人员的兴趣和参与，已成为国内外学术界十分热门的研究课题. 由于 GA 与 ANN 的来源不同，GA 是从自然界生物进化机制获得启示的，而 ANN 则是基于人脑神经网络若干基本特性的抽象和模拟. 因此，它们在信息处理方式上存在着较大差异. 近年来，已有越来越多的研究人员尝试着将 GA 与 ANN 相结合，希望充分利用两者的长处，以找出一种有效的解决问题的方法.

已有的研究结果表明，遗传算法在早期阶段能非常迅速地收敛到一个近似解. 但由于 GA 技术不适宜局部搜索，所以到后来收敛速度明显下降. 鉴于此，使用一种将遗传算法与 BP 算法相结合的关于结构化神经网络的混合训练方法，在训练初始阶段采用 GA 技术，然后采用 BP 算法来训练网络，以便提高整体训练速度. 下面结合训练过程来介绍这一方法.

1. 编码与群体初始化

采用实数编码方案，每个连接权值直接用一个实数表示，网络权值的一种分布用一组实数(称为一个个体)来表达. 设初始群体由 N 个个体组成，并对其实施单一化处理，即把相同的个体单一化，不允许群体中有若干个相同个体出现.

2. 适应度函数的确定

适应度函数的确定方法很多，采用：

$$F(E) = E_{\max} - E \tag{3.7.15}$$

其中 $E_{\max}$ 是误差函数的可能最大值，误差函数定义为：

$$E = \sum_m \sum_k (Y_{mk} - \overline{Y}_{mk})^2 \tag{3.7.16}$$

其中 Y_{mk} 及 $\overline{Y}_{mk}$ 分别为第 m 个训练样本的第 k 个输出节点的实际输出与期望输出，其计算过程包括以下几步：

(1)把某一个学习模式的值作为输入层单元的输出 I_i,用输入层到隐层的权值 W_{ij} 和中间层单元的阈值 θ_j 求出中间层单元 j 的输出 H_j:

$$H_j = g(\sum_i W_{ij} I_i - \theta_j) \tag{3.7.17}$$

(2)用中间层的输出 H_j,中间层到输出层的权值 V_{jk} 以及输出层单元 k 的阈值 γ_k 求出输出层单元 k 的输出:

$$Y_k = g(\sum_{j=1}^{l} V_{jk} H_j - \gamma_k) \tag{3.7.18}$$

其中 l 为隐层单元的数目;g 为神经元的激活函数,取为:

$$g(x) = \frac{1 - e^{-\beta x}}{1 + e^{\beta x}} \tag{3.7.19}$$

(3)由学习模式的教师信号和输出层的输出得到第 m 个样本模式的第 k 个节点的误差 e_{mk}:

$$e_{mk} = Y_{mk} - \overline{Y}_{mk} \tag{3.7.20}$$

而

$$E = \sum_m \sum_k e_{mk}^2$$

如(3.7.16)式所示.

3. 选择继承

评价各个权值及阈值,对适应度为 F_i 的个体赋予其选择概率 p_i:

$$p_i = F_i / \sum_{j=1}^{N} F_j \tag{3.7.21}$$

在实际训练中,一般将适应度最大的个体直接遗传给下一代.

4. 交叉和变异

采用实数编码时,以变异概率 p_m 随机地改变串中的某些位.在自适应遗传算法中,修正了在运用交叉和变异算子时通常所使用的方法.首先,若新一代的最佳个体明显不优于前一代的最佳个体,则将交叉概率变为 1.因此,在这种情况下,新一代中每一个体都要执行交叉操作.其次,变异方法不再是通常的将所选定的个体的值简单地改变为某一新的随机值,也不是以同等的机会按相同的比例来修正所选定的个体的当前值.

作为对于变异方法的改进,提出了一种自适应变异操作,现概述如下:

设 t_1, t_2 为程序运行的两个时刻,E_1, E_2 分别为对应于 t_1, t_2 时刻的最佳个体的误差函数值.定义相对误差为

$$E_r = \left| \frac{E_2 - E_1}{E_1} \right| \tag{3.7.22}$$

设所选择的个体为 S,它的第 i 位发生变异.记变异后的个体为 S',规定其变异按如下方式进行:

$$S'(i) = S(i) + \text{Random}(2) \cdot \delta \cdot \lambda \cdot (1 - T/T_{\max}) \tag{3.7.23}$$

其中

$$\delta = \begin{cases} -1 & \text{当 Random}(2) = 0 \\ 1 & \text{当 Random}(2) = 1 \end{cases}$$

T为迭代次数，T_{max}为预先规定的最大迭代次数，Random 是随机数生成器. 为了加快收敛速度，同时采用两种途径来控制变异强度. 首先是在迭代的前一阶段取较大的变异系数$\lambda=\lambda_1$，而在迭代一段时间后，当由(3.7.22)式定义的相对误差连续若干次下降不明显时，适当减小变异系数，即若$E_r<\varepsilon_0$并且$t_2=t_1+T_0$，则变异系数取为$\lambda=\lambda_2$，其中$\lambda_1>\lambda_2$. 一般情况下，$\lambda\in[0.1, 0.3]$. 取$\lambda_1=0.2$，$\lambda_2=0.1$，ε_0和T_0为预先设定的常数，取$\varepsilon_0=0.05$，$T_0=10$. 除此之外，变异强度还按(3.7.23)式随迭代次数的增加而逐渐减小. 利用这种方法，由于无论在正向还是反向上，个体的值的改变都相对较小，而且变异强度还随迭代次数的增加而逐渐减小，这样就使得在遗传算法运行后期，变异操作丧失较好的染色体基因这一问题能够得以缓解. 模拟实验结果表明，这种改进的自适应变异操作明显优于定步长变异操作，其收敛速度大约可平均提高10%～20%.

5. 重复或终止

从当前父代和子代的各种取值中重新排序选择出N个适应度值较高的个体作为下一代的样本. 转步骤2重新进行训练，终止条件为群体适应度趋于稳定，或误差小于某一给定值，或运算已达到预定的进化代数.

6. 遗传算法与 BP 算法的自适应切换

鉴于遗传算法在早期阶段能非常迅速地收敛到一近似解，而在后期阶段其收敛速度明显下降这一特点，提出了一种在训练初始阶段采用 GA 技术，然后采用 BP 算法来训练网络，以便提高整体训练速度的遗传进化神经网络方法. 在算法运行过程中将遗传算法与 BP 算法进行自适应切换的实施方案如下：

仍采用由(3.7.22)式定义的相对误差. 若相对误差连续若干次下降不明显，则进行遗传算法与 BP 算法的切换，即若$E_r<\varepsilon_1$并且$t_2=t_1+T_1$，则跳出遗传算法训练过程，进入 BP 算法训练过程. 其中ε_1和T_1为预先设定的常数，取$\varepsilon_1=0.01$，$T_1=50$.

7. 执行 BP 算法：

此处略去 BP 算法的执行步骤. BP 算法的运行终止条件为误差E小于某一给定值，或运算已达到预定的迭代次数.

为进一步说明遗传进化神经网络方法，下面给出算法的流程图. 为使算法流程主要走向清晰，图 3.42 中仅标出了误差函数值达到指定要求$E<\varepsilon$时的程序运行终止出口，其中ε为预先设定的正的小数.

三、算例

本节以缓冲包装材料中常见的具有三次非线性和双曲正切非线性的情形为例，说明用遗传算法进化结构化神经网络这一方法对于典型的缓冲包装材料非线性特性识别问题的有效性，并与 BP 算法进行了比较. 在模拟实验中首先假设非线性项$p(x)$为一已知函数，由数值方法获得实验数据，利用这些数据训练网络，然后由训练好的网络识别出系统的非线性特性. 算例中取激振力$f(t)=f_0\sin(314t)$，隐层节点数$l=5$，群体规模$N=200$. 最大迭代次数$T_{max}=20000$，变异概率$p_m=0.1$，初始可行解群体由随机法产生.

算例 1：假设$p(x)=k_1x^3$，即为三次非线性型. 选用模拟参数$c/m=1000$，$k/m=1.2\times10^5$，$k_1/m=10^5$，$f_0/m=5\times10^4$，$\Delta t=0.001$ s. 采用常规 BP 算法，取训练速率$\eta=0.5$，惯性系数$\alpha=0.0$. 经过 650 次训练后，误差小于 0.4. 采用 GA 进化神经网络方法，经过 350 次训练后，误差即可小于 0.4. 但是同时经过 3000 次运算后，采用常规 BP 算法，误差可达 9.007×

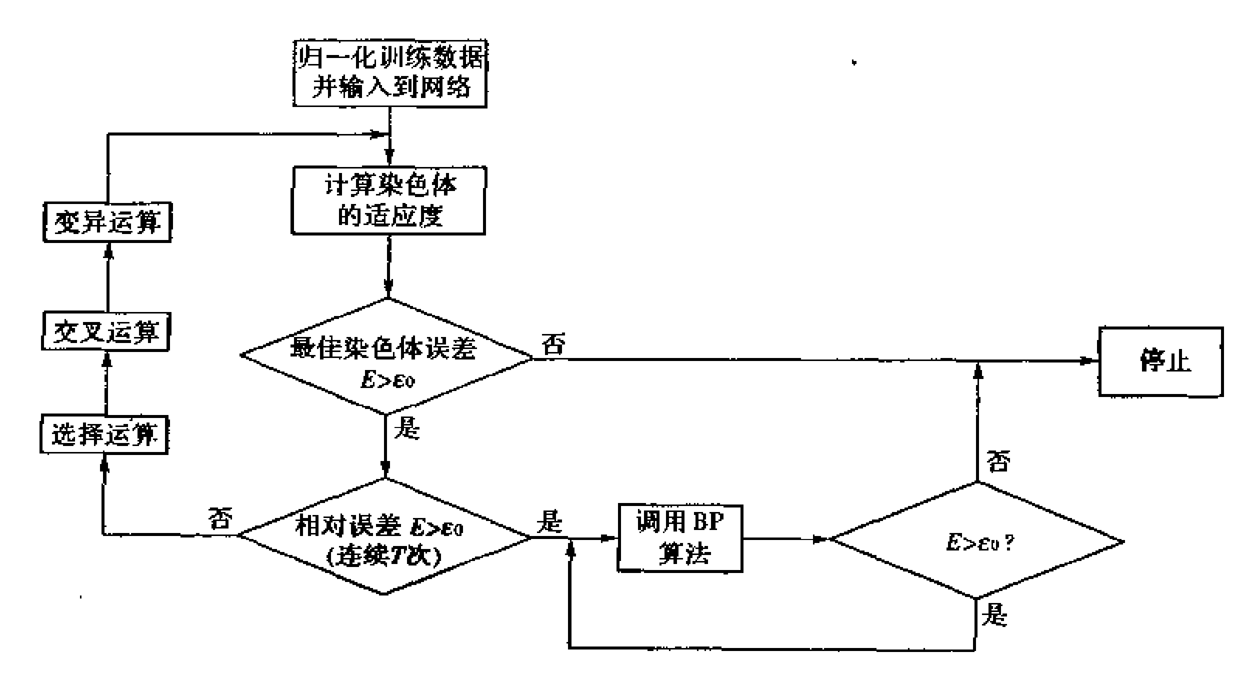


图 3.42　遗传进化神经网络混合训练流程图

10^{-2}，而采用 GA 进化神经网络方法，误差仅达 1.442×10^{-1}. 由此例可见，采用 GA 进化神经网络方法训练在早期阶段收敛速度要快于 BP 算法，但由于遗传算法不适宜局部搜索，所以到后来反而要比 BP 算法收敛得慢. 对于这类非线性特性识别问题，可采用在训练初始阶段利用进化方法，然后采用 BP 算法的混合型训练方法，这样可有助于整体训练速度的提高.

算例 2：假设 $p(x)=\gamma\tanh(kx/\gamma)$，即为双曲正切非线性型，式中 γ 为弹性力的渐近值. 许多较理想的缓冲材料，如瓦楞纸、蜂窝板、聚本乙烯泡沫塑料等，都具有这类特性. 选用模拟参数为：$c/m=40$，$k/m=1000$，$\gamma=100$，$f_0/m=250$，$\Delta t=0.003$ s. 采用常规 BP 算法，取训练速率 $\eta=0.4$，惯性系数 $\alpha=0.0$，经过 600 次训练后，误差小于 1.1×10^{-3}. 采用进化神经网络混合方法，经过 600 次训练后，误差可小于 1.68×10^{-4}. 训练 1000 次后，误差可达 9.5×10^{-5}. 对于误差等于 9.5×10^{-5} 的情形，绘制了识别出的非线性与实际非线性特性的比较图(图 3.43).

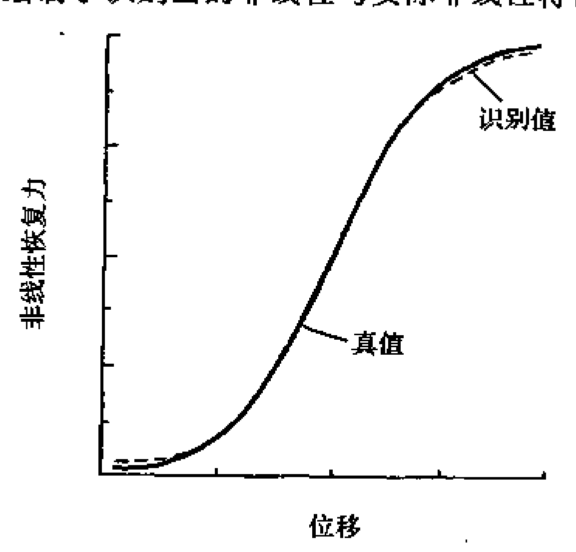


图 3.43　非线性特性识别结果与实际值比较

由图 3.43 可见，对于典型的非线性缓冲包装材料模型，根据进化神经网络混合方法获得的非线性特性的识别结果与相应的实际值吻合良好.

§3.8 自适应线性元件

自适应线性神经元(Adaline)是在 1961 年由斯坦福大学教授 Windrow 提出的，它是一个自适应可调的网络，适用于信号处理中的自适应滤波、预测和模型识别. 构成这种网络的基本单元 Adaline 是一个类似于感知器单元，如图 3.44 所示. 如在第 k 时刻，有向量 x_k 输入，权向量为 w_k，此时有模拟输出 y_k，$y_k=w_k^{\mathrm{T}}x_k$ 及二进制输出 q_k. y_k 与要求的理想响应的差值，通过

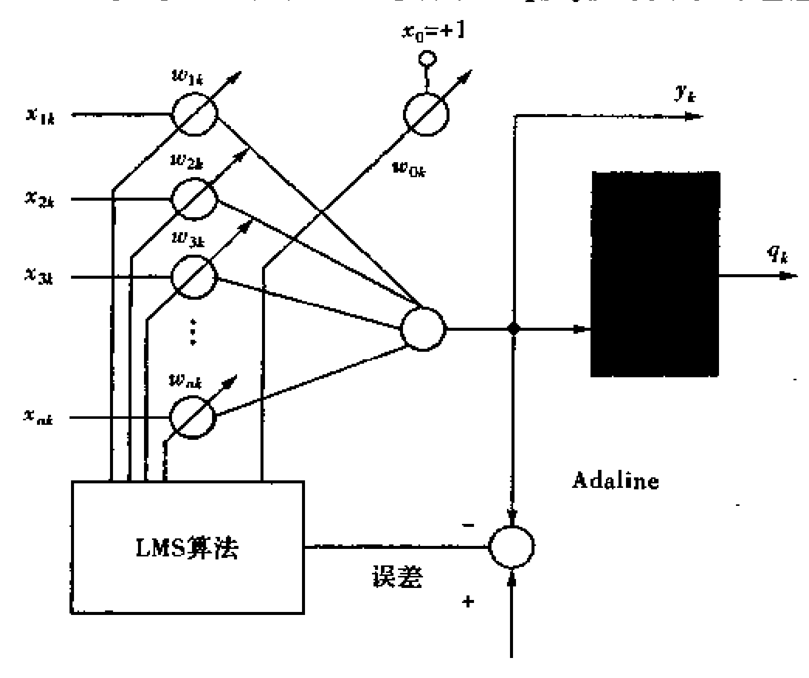


图 3.44 Adaline 原理图

LMS 算法，修改 w_k 为 w_{k+1}，从而减小了 y_k 与理想响应的误差. 这个单元的输入与输出关系满足：

$$y_k=\sum_{i=1}^{n}w_{ik}x_{ik}-\theta_k \tag{3.8.1}$$

$$q_k=\mathrm{sgn}(y_k) \tag{3.8.2}$$

$x_k, w_k\in\mathbf{R}^n$，θ_k 为阈值，如使 $\theta_k=w_{0k}$，$x_0=1$，那么

$$y_k=\sum_{i=0}^{n}w_{ik}x_{ik} \tag{3.8.3}$$

比较(3.8.1)式与(3.8.2)式，其差别只是输出分为模拟和数字两个部分. 从数字部分看，与感知器单元完全相同，它可进行线性分割. 从模拟输出看，它是作为误差调节之用，对单个 Adaline，其误差为模拟输出和要求响应输出之差，也为模拟量. 用 LMS 算法能保证这种网络在自适应学习时的收敛性. Adaline 的特点是可以根据外界输入的情况和要求的响应随时学习，十分灵活.

一、用于非线性分割的自适应网络

对于单层的 Adaline 网络，由于它的形式与感知器相同，因而它只能适用于一种线性分割的情况，为了要达到非线性的分割，可采用下列两种方法：

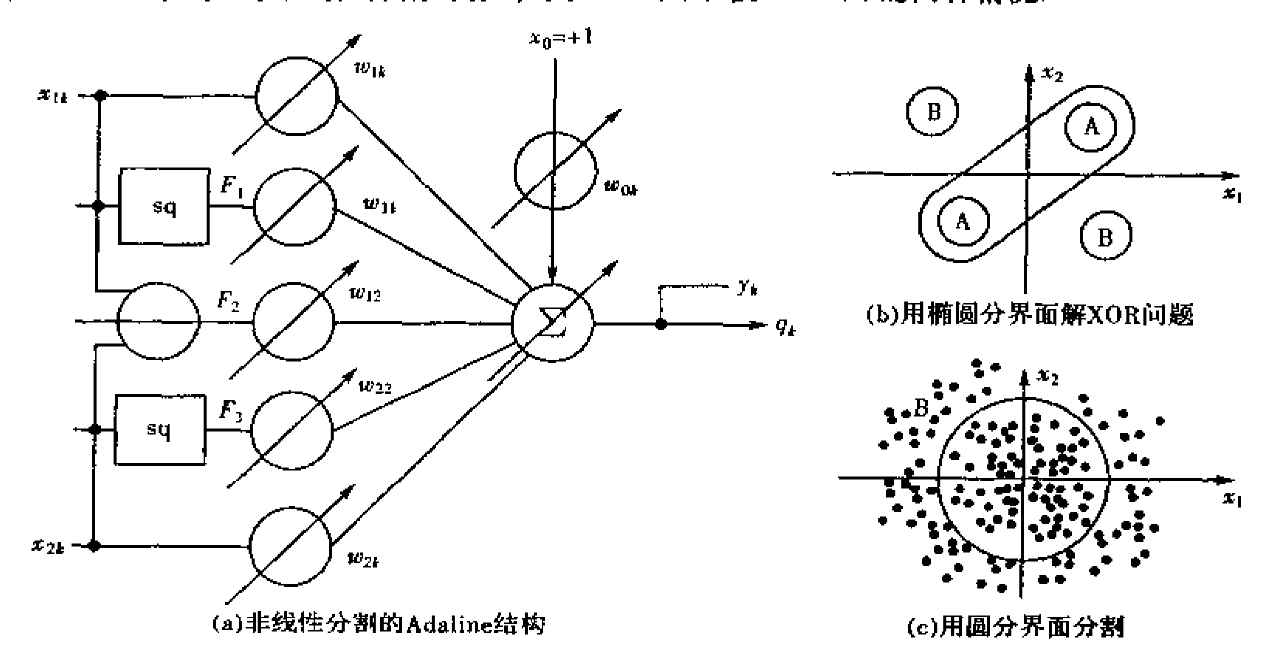
(1)加一些辅助的输入单元，使输入向量 $x_k\in\mathbf{R}^n$ 变为 $x_k\in\mathbf{R}^{n+l}$，这 l 个输入单元的输入是

x_k 矢量中各单元 x_{ik} 的二次值，$i=1,2,\cdots,n$. 如在两维的情况下，$n=2$，我们加上三个辅助输入节点 F_1、F_2、F_3，如图 3.45(a)所示，使其 $F_1=x_{1k}^2$，$F_2=x_{1k}x_{2k}$，$F_3=x_{2k}^2$. 这样我们得到其输出 y_k 为

$$y=\sum_{i=0}w_{ik}x_{ik}+w_{11}F_1+w_{12}F_2+w_{22}F_3=$$

$$w_{0k}+x_{1k}w_{1k}+x_{2k}w_{2k}+w_{11}x_{1k}^2+w_{22}x_{2k}^2+w_{12}x_{1k}x_{2k} \quad (3.8.4)$$

当 $y_k=0$ 时，在二维空间上的分界面由(3.8.4)式给出，是一个椭圆或是一个圆，可以通过调节权系数来达到这一类的非线性分割，例如在图 3.45(b)和图 3.45(c)的两种情况.



(a)非线性分割的Adaline结构

(b)用椭圆分界面解XOR问题

(c)用圆分界面分割

图 3.45

当然，这种辅助输入单元的网络将会因为输入维数的增加，其辅助输入维数亦增加，如原来输入单元维数为 n，考虑二次情况，输入总的维数为：

$$2n+C_n^2=2n+\frac{n(n-1)}{1\times 2}=\frac{n^2+3n}{2}$$

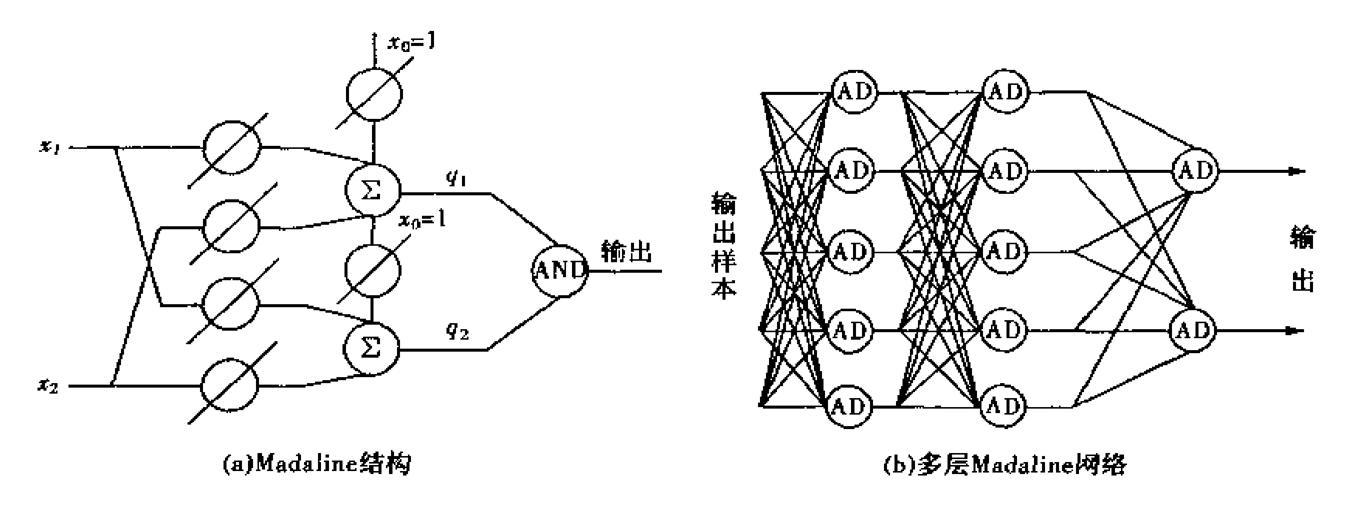
因而增加了复杂性.

(2)Madaline 两络，是由多个 Adaline 单元组合成的多层网络，圆 3.46(a)为两个 Adaline 和一个与门组合成的 Madaline 网络，多个 Adaline 组合成的多层两络如图 3.46(b)所示，图中 AD 是表示 Adaline 的符号.

它的非线性分割与多层感知器一样，如输入矢量为 n 维，利用组合很多的 $n-1$ 维的超越平面来实现.

二、多层 Adaline 网络的学习算法

多层 Adaline 的学习分两部分进行，先利用 MRII 的算法计算出每一层的输出要求，然后再采用单层的 LMS 法来完成每一层间的权的学习. 如果输入是一个模拟量，第一层的 Adaline 的输出是两个量，一是模拟量 y_k，提供给 LMS 算法用来调整该层的辅入权，另一是数字量 q_k，是输入到下一层的网络. 这样从第二层 Adaline 开始，其输入矢量是二进制的量，MRII 算法就是建立在这些二进制量的基础上. 它的思想是：从对网络干扰最小的二进制输出量，改变其符号，如改变第一层中某一个 Adaline 输出的符号，(从$+1\rightarrow-1$，或从$-1\rightarrow+1$)观察其总



(a)Madaline结构　　　　　　(b)多层Madaline网络

图 3.46　Madaline 结构和多层 Madaline 网络

的网络的输出响应和要求的值之间的误差，如误差减小，就确认这种改变，若误差没有减小，则恢复原来符号，每次选中的 Adaline 应是那些模拟量最接近 0 的单元，这样可使网络的干扰最小.

这样我们可以列出 MRII 的学习步骤：

(1)初始化多层 Adaline 网络中的权，用一个随机数作为各个权的初始权值.

(2)输入一个样本矢量 x_k 和要求的输出 t_k 用式(3.8.1)按层一步一步地计算出实际输出，并求出要求的输出 t_k 和实际输出的误差.

(3)根据最小干扰的原则，从第一层开始，找第一层中神经元模拟输出最接近于 0 的那个神经元，让它输出数字量改变符号.

(4)计算网络最后的输出和要求响应之间的误差，如果误差减小，则这种改变被接受，如没有减小，上一步改变的输出数字量的符号恢复.

(5)第一个神经元训练结束后，转入下一个神经元，即模拟输出第二个接近 0 的单元，仍按上面的规则训练.

(6)当这一层单个神经元训练完了，再按两个一组地训练，然后再按三个一组，……直到输出与要求响应之间的距离减到最小为止. 然后用同样的方法训练第二层、第三层，直到相同量为止.

(7)另一个新的样本输入，用同样方法进行训练，一直到达到误差最小为止.

(8)通过 MRII 的学习可以得到每个样本输入对应的每一层相应的输出，从而对每一单层的 Adaline 可以用 LMS 算法进行学习，得到全部权的解.

单层的学习算法如下：

$$w(t+1) = w(t) + \frac{\alpha}{\|x(t)\|^2}\varepsilon(t)x(t) \tag{3.8.5}$$

这里，$w(t+1)$为 $w(t)$的下一个时刻的权矢量，$x(t)$为当前输入样本矢量，$\varepsilon(t)$为当前的误差. t 为迭代的次数.

如果 $x(t)$矢量的分量为±1，那么 $\|x(t)\|^2$ 为：

$$\|x(t)\|^2 = x_1^2(t) + x_2^2(t) + \cdots + x_{n+1}^2(t) = n+1$$

对于每个调节周期，可按照(3.8.5)进行调节. 误差 $\varepsilon(t)$为：

$$\varepsilon(n_0) = t(n_0) - x^{\mathrm{T}}(n_0)w(n_0) \tag{3.8.6}$$

当权改变时，其误差改变为：

$$\Delta\varepsilon(t) = \Delta\lfloor t(t) - \boldsymbol{x}^{\mathrm{T}}(t)w(t)\rfloor = -\boldsymbol{x}^{\mathrm{T}}(t)\Delta w(t) \tag{3.8.7}$$

$$\Delta w(t) = w(t+1) - w(t) = (\alpha/(\|\boldsymbol{x}(t)\|^2))\varepsilon(t)\boldsymbol{x}(t) \tag{3.8.8}$$

综合(3.8.7)式和(3.8.8)式得

$$\Delta\varepsilon(t) = -\boldsymbol{x}^{\mathrm{T}}(t)\frac{\alpha}{\|\boldsymbol{x}(t)\|^2}\varepsilon(t)\boldsymbol{x}(t) = -\alpha\,\varepsilon(t) \tag{3.8.9}$$

这说明，当网络的输入不变时，误差的变化缩小当前误差的 α 倍，选择 α 因子可以控制收敛的速度和稳定性，α 太大，误差将可能增大．一般取 $1.0>\alpha>0.1$．

三、Adaline 和 Madaline 的应用

单个 Adaline，作为调节单元可以形成一个自适应滤波器，用于信号处理．图 3.47 是一个 Adaline 单元，它的输入构成一个随时间变化的信号，输入的信号通过延迟得到 $x_{k-1},\cdots,x_{k-l}$，l 为延迟器的最大数目，它们是通过对输入信号的采样而得到的．因为信号随时间变化，因而当前采样值为 x_k，在 Adaline 的输入端分别为 $x_k,x_{k-1},\cdots,x_{k-l}$，下一时刻为 $x_{k+1},x_k,\cdots,x_{k-l+1}$，$x_{k+1}$ 为当前时刻，x_{k-l+1} 为延迟了 l 个节拍后的采样值，即上 l 个节拍前的采样值．Adaline 网络输入 $l+1$ 维的变化的信号．自适应滤波的目的是通过要求的响应和实际输出的误差，调节其权而达到对输入信号滤波的目的(见图 3.48)．

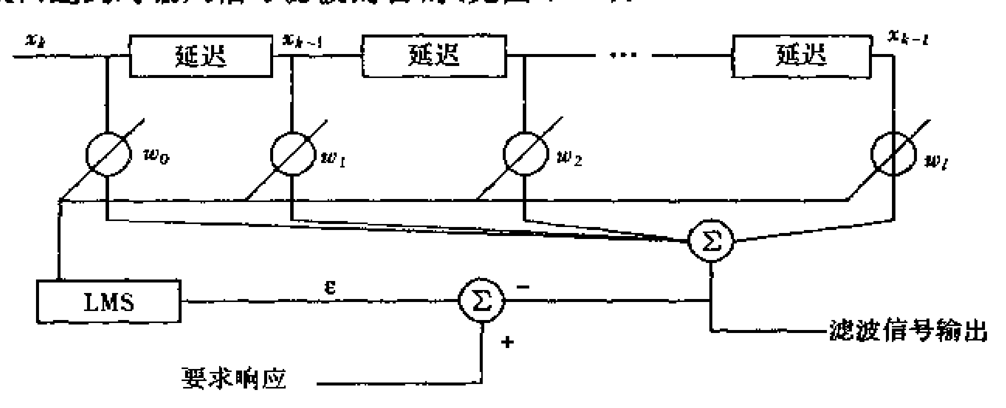


图 3.47　Adaline 组成的自适应滤波器

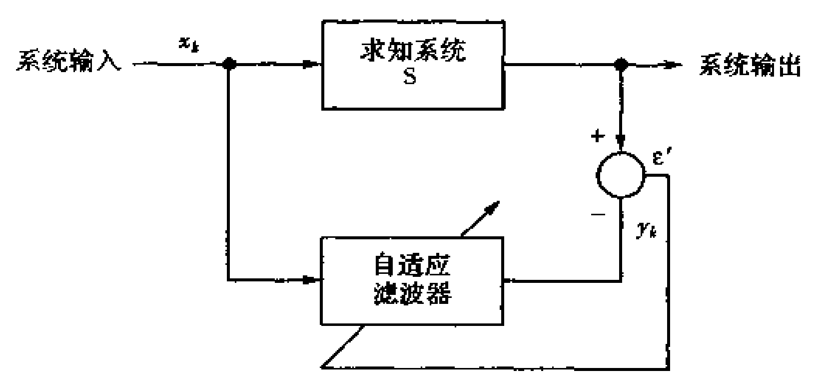


图 3.48　用自适应滤波器进行建模的模图

§3.9　径向基函数神经网络

除上面讲的作用函数用 Sigmoid 函数的网络外还有一种常用的网络，即径向基函数网络．下面我们从几方面来了解这种函数网络．

3.9.1　网络结构

径向基函数神经网络结构见图 3.49 所示，它只有一个隐层，输出单元是线性求和单元，即输出是各隐单元的加权求和. 隐单元的转移函数用径向基函数(Radial Basis Function，RBF)，输入单元和隐单元的连接权值固定为 1，只有隐单元和输出单元的连接权值为可调.

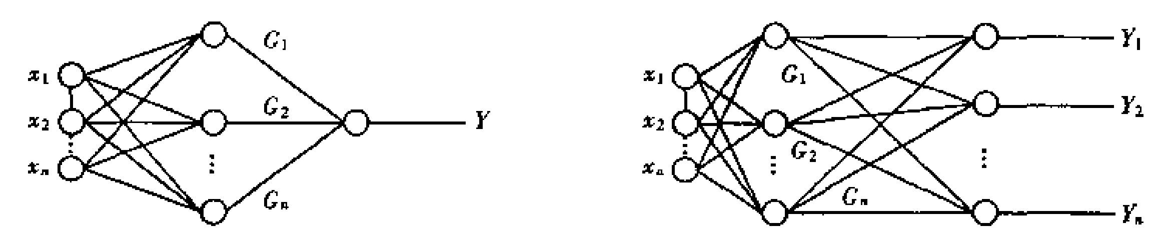


图 3.49　径向基函数神经网络结构

可以把 RBF 看成对未知函数 $F(x)$的逼近器，一般任何函数都可以表示成一组基函数的加权和，这相当于选择各隐单元的作用函数构成一组基函数用于近似 $F(x)$. 由模式识别理论可知，在低维空间非线性可分问题总可以映射到一个高维空间，使其在高维空间中为线性可分. 在 BRF 网络中，输入单元到隐单元的映射为非线性的(隐单元的转移函数为非线性的)，而隐单元到输出单元则是线性的. 因而可以把输出单元部分看做一个单层感知器，这样，只要适当选择隐单元数(高维空间的维数)及其作用函数，就可以把原来的问题映射为一个线性可分问题，从而可用一个线性单元解决问题.

3.9.2　网络算式及参数

最常用的径向基函数形式为高斯函数，它的可调参数有两个，即中心位置及方差 b(函数的宽度参数)，用这类函数时整个网络的可调参数(待训练的参数)有三组，即各基函数的中心位置、方差和输出单元的权值. 通常选择隐层的节点数为训练样本个数，每个节点都有一个径向基函数的中心向量，该中心向量即为训练样本的输入向量.

$$C_K = \{C_{K1}, C_{K2}, \cdots, C_{Kn}\} \qquad K = 1,2,\cdots,N \tag{3.9.1}$$

隐节点的净输入定义为输入模式 $\boldsymbol{X}$ 与隐节点的径向基函数中心向量间的欧氏距离，即：

$$\delta_K = \| \boldsymbol{X} - \boldsymbol{C}_K \|^2 = \sum_{i=1}^{n} (x_i - C_{Ki})^2 \qquad K = 1,2,\cdots,N \tag{3.9.2}$$

隐层节点的转移函数为 Gauss 函数：

$$f(x) = \exp(-x^2/b) = \exp(-\delta_K^2/b) \tag{3.9.3}$$

Gauss 函数其形状见图 3.50 所示，其中参数 b 控制钟形高斯曲线宽度的作用，隐层节点的输出 $Z_K = f(\delta_K)$代表着输入模式离开该隐节点所代表的径向基函数中心的程度. 输出层节点数为输出向量的维数，节点 j 的输出为：

$$\boldsymbol{Y}_j = \sum_{K=1}^{N} w_{kj} z_K = \boldsymbol{W}_j \boldsymbol{Z} \tag{3.9.4}$$

式中

$$\boldsymbol{W}_j = [w_{1j}, w_{2j}, \cdots, w_{Nj}], \qquad \boldsymbol{Z} = [z_1, z_2, \cdots, z_N] \tag{3.9.5}$$

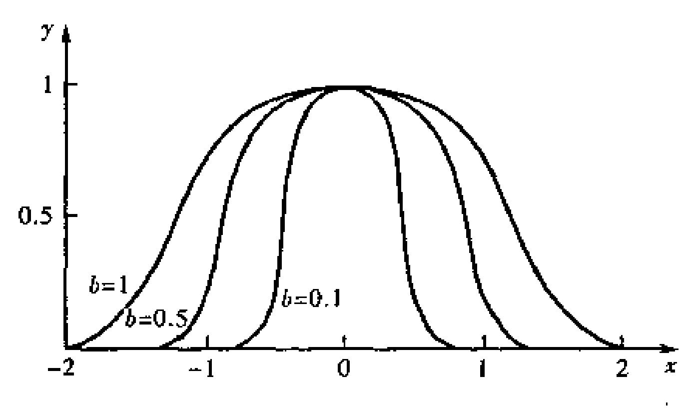


图 3.50　Gauss 函数

RBF 网络的中心向量和权值均由学习样本来确定，因输出单元是线性单元，所以它的权值可以用最小二乘法直接计算出来. 因此，设计一个 RBF 网络时只有一个参数 b 需要调整，即高斯函数中的平滑因子 b，它控制着高斯曲线钟形的宽度.

RBF 网络的中心向量、平滑因子 b 和权值 w 也可由 BP 算法学习训练得到.

第四章 反馈型神经网络

§4.1 概 述

4.1.1 前馈型与反馈型神经网络的比较

前馈型与反馈型神经网络的比较如下：

(1)前馈型神经网络取离散或连续变量，一般不考虑输入和输出之间的延迟，只表达输入输出之间的映射关系.反馈型神经网络也可取离散或连续变量，但是，考虑输入输出之间在时间上的延迟，因此，需要用动态方程来描述.前馈型神经网络仅实现非线性映射，而反馈型神经网络是一个非线性动力学系统.

(2)前馈型神经网络的学习训练主要采用 BP 算法，计算过程和收敛速度比较慢.而反馈型神经网络的学习主要采用 Hebb 规则，一般情况下计算的收敛速度很快，并且它与电子电路有明显的对应关系，使得网络易于用硬件实现.

(3)前馈型神经网络学习训练的目的是快速收敛，一般用误差函数来判定其收敛程度.反馈型神经网络的学习目的是快速寻找到稳定点，一般用能量函数来判别是否趋于稳定点.

(4)两者都有局部极小问题.

4.1.2 反馈型神经网络模型

一、网络结构

反馈型神经网络的结构如图 4.1 所示，首先考虑单层全反馈型网络，这种网络的每个节点的输出都和其他的节点的输入相连，输入输出关系为：

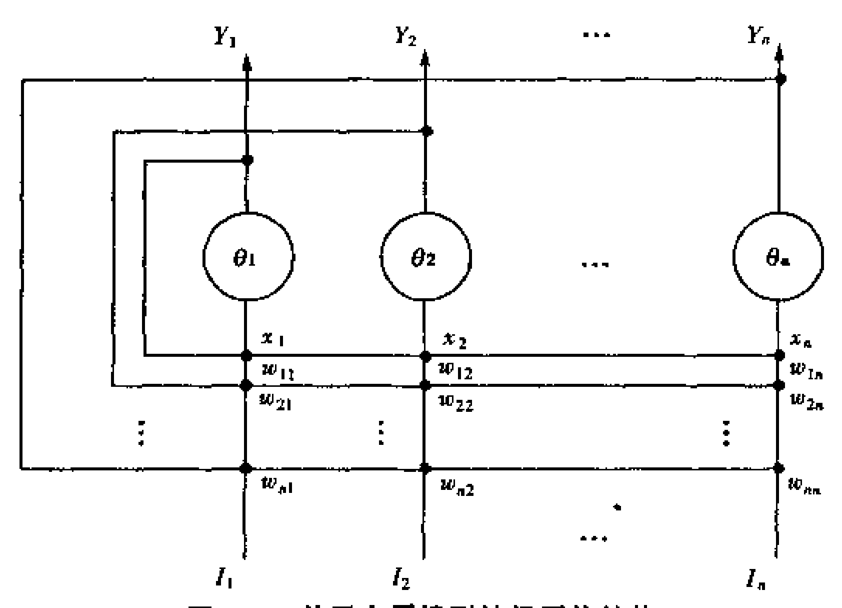


图 4.1 单层全反馈型神经网络结构

$$Y_j = f(x_j) = f(\sum_{i=1}^{n} w_{ij} Y_i + I_j - \theta_j), \quad j = 1,2,\cdots,n \tag{4.1.1}$$

二、网络状态

对于有 n 个节点组成的反馈网络，在某一时刻 t，其状态矢量为 $\boldsymbol{X}=(x_1,x_2,\cdots,x_n)$，$\boldsymbol{X}\in\mathbf{R}^n$，输出矢量为 $\boldsymbol{Y}=(y_1,y_2,\cdots,y_n)$，$\boldsymbol{Y}\in\mathbf{R}^n$，可以用 $\boldsymbol{X}(t)$ 和 $\boldsymbol{Y}(t)$ 来表示网络的状态. 由 $\boldsymbol{Y}(t)$ 可以得下一时刻的 $\boldsymbol{X}(t+1)$，而 $\boldsymbol{X}(t+1)$ 又引起 $\boldsymbol{Y}(t+1)$ 的变化，这种不断的反馈演化过程，使得状态矢量 $\boldsymbol{X}(t)$ 随时间变化. 在一个 n 维的状态空间上，状态的变化过程可以用一条轨迹来描述，从初始状态 $\boldsymbol{X}(t_0)$ 出发，$\boldsymbol{X}(t_0+\Delta t)\rightarrow\boldsymbol{X}(t_0+2\Delta t)\rightarrow\cdots\rightarrow\boldsymbol{X}(t_0+m\Delta t)\rightarrow\cdots$，这些在空间上的点所确定的轨迹，是演化过程中所有可能状态的集合，并称为状态空间.

对于不同的权值 w_{ij} 和输入 $I_i(i,j=1,2,\cdots,n)$，网络的状态轨迹可能出现以下几种情况：

(1)轨迹经过一段时间 $t(t>0)$ 后不会再延伸，而永远停留在 $\boldsymbol{X}(t_0+t)$ 状态，这时称网络收敛到一个**稳定点或平衡点**. 由于 $\boldsymbol{X}(t_0+t)$ 不再变化，所以相对应的 $\boldsymbol{Y}(t_0+t)$ 也达到了稳定值. 对于非线性系统来说，不同的初始状态 $\boldsymbol{X}(t_0)$，可能会有不同的轨迹，到达不同的稳定点. 在一个反馈网络中，可能存在有多个稳定点，根据不同的情况，这些稳定点可分为：

①渐近稳定点 $\boldsymbol{X}_e$. 如果在稳定点 $\boldsymbol{X}_e$ 周围的 $s(\sigma)$ 区域内，从任一初态 $\boldsymbol{X}(t_0)$ 出发，当 $t\rightarrow\infty$ 时都收敛于 $\boldsymbol{X}_e$. 这样它不仅存在一个稳定点 $\boldsymbol{X}_e$，而且存在一个稳定区域，有时也称这个稳定点为吸引子，对应的稳定区域称为吸引域.

②不稳定的平衡点 $\boldsymbol{X}_f$. 在某些特定的轨迹演化过程中，能够使网络达到稳定点 $\boldsymbol{X}_f$，但是对于 $\boldsymbol{X}_f$ 的其他方向上，如果有一个小的区域 $s(\sigma)$，不管 $s(\sigma)$ 取多么小，其轨迹在时间 t 以后总是偏离 $\boldsymbol{X}_f$.

③网络的解. 如果网络最终稳定在设计人员所期望的稳定点上，而且该稳定点又是渐进稳定点，那么这个稳定点就是网络的解.

④网络的伪稳定点. 如果网络最终稳定在一个渐进稳定点上，但是这个稳定点不是网络设计所要求的解，则这个稳定点为伪稳定点.

在一个非线性反馈的网络系统中存在着不同类别的稳定点，而网络设计的目的是希望网络能稳定在所要求的稳定点上，并且还要有一定的稳定域.

(2)轨迹为环状，称为**极限环**. 在某些参数条件下，$\boldsymbol{X}(t)$ 的轨迹可能是一个圆，或为一个环，状态 $\boldsymbol{X}(t)$ 沿环重复旋转，永远不会停止，这时 $\boldsymbol{Y}(t)$ 也出现周期性变化，说明系统产生了振荡. 如果轨迹的变化是在在两种状态下振荡，其极限环为 2，如果在 m 种状态下振荡，其极限环为 m.

(3)如果 $\boldsymbol{X}(t)$ 的轨迹在某个确定的范围内变化，但既不重复又不能停下来，状态变化为无穷多个，而轨迹也不发散到无穷远，这种现象成为混沌(Chaos).

(4)如果 $\boldsymbol{X}(t)$ 的轨迹随时间一直延伸到无穷远，此时状态发散，而系统的输出也发散. 但是在神经网络中，输入输出关系为一个有界函数，虽然状态是发散的，但是其输出 $\boldsymbol{Y}(t)$ 还是有界的，而 $\boldsymbol{Y}(t)$ 的有界性又限制了状态的发散.

对于 n 个神经元组成的反馈系统，它们的行为是由这些状态轨迹所决定的，目前神经网络是用第一种情况来解决某些问题，如果把系统的稳定点作为一个记忆的话，那么从初态朝这个稳定点流动的过程就是寻找记忆的过程. 初态可以看做是给出的部分记忆信息，状态 $\boldsymbol{X}(t)$ 流动的过程是从部分信息寻找全部信息，这就是联想记忆的过程. 如果把系统的稳定点考虑为一

个能量函数的极小点，在状态空间中，从初态 $X(t_0)\to X(t_0+t)$，最后达到 X^*，若 X^* 为稳定点，则可以看做是 X^* 把 $X(t_0)$ 吸引过来. 在 $X(t_0)$ 能量较大，而吸引到 X^* 能量比较小了，那么能量的极小点就可以作为一个优化目标函数的极小点，状态变化的过程就是优化一个目标函数的过程. 因此反馈网络的状态流动就是一种联想记忆或优化的过程，如果适当选择连接权值就可以达到这个目的.

三、网络的设计要求

对反馈型神经网络的权值和阈值的设计需要考虑以下三个方面：

(1)网络的稳定性. 要求网络系统能够达到稳定收敛，避免出现振荡和混沌现象.

(2)网络的稳定点. 一个非线性网络可能有多个稳定点，权值的设计要求某些稳定点是所要求的解. 对于联想记忆的网络，希望每个稳定点都是一个记忆，那么记忆容量就和稳定点的数量有关，希望记忆的量越大，就要求稳定点的数越多. 但是稳定点的数目越多，会引起吸引域的减小，从而使联想记忆功能减弱. 另外，能量函数往往只有一个全局最小点，那么稳定点越多，陷入局部极小的可能性越高.

(3)稳定点的吸引域. 使期望解的吸引域尽可能大，而非期望解的吸引域尽可能小.

§4.2 离散型 Hopfield 神经网络

美国加州工学院物理学家 Hopfield J J 提出的一种网络，称为 Hopfield 网络(Hopfield Neural Network，简称 HNN). HNN 是目前研究最充分、应用最广泛的反馈型神经网络模型. Hopfield 将能量函数的概念引入分析反馈型神经网络的稳定过程，使得网络运行的稳定性判断有了可靠的依据. 由于 HNN 与电子电路存在明显的对应关系，其易于用硬件实现，为神经网络计算机的研究开发奠定了基础. 同时，HNN 也开辟了神经网络应用于联想记忆和优化计算的新领域.

HNN 又分为两种模型，一种是离散的随机模型，称为离散型 Hopfield 神经网络(Discrete Hopfield Neural Network，简称 DHNN)；另一种是连续模型，称为连续型 Hopfield 神经网络(Continuous Hopfield Neural Network，简称 CHNN).

4.2.1 离散型 Hopfield 神经网络模型

一、网络结构

离散型 Hopfield 神经网络(DHNN)是一个离散的时间系统，它可以用一个加权的无向图表示，图的每一个边都附有一个权值，图的每一个节点都有一个阈值，网络的阶数相应于图中的节点数.

DHNN 的结构见图 4.1 所示，这是一个单层结构的全反馈网络，有 n 个节点，$\boldsymbol{W}$ 是一个 $n\times n$ 的对称零对角权值矩阵，$\boldsymbol{\theta}$ 为 n 维阈值向量. 每个节点可处于两个可能的状态之一，即 1 或 −1. 假设各节点的外加输入 $I_i=0, i=1,2,\cdots,n$. 令 $X_i(t)$ 表示 t 时刻节点 i 的状态，则节点 i 的下一个状态由下面算式决定：

$$X_i(t+1)=\operatorname{sgn}(H_i(t))=\begin{cases}1 & H_i(t)\geqslant 0\\ -1 & H_i(t)<0\end{cases} \tag{4.2.1}$$

$$H_i(t)=\sum_{j=1}^{n}w_{ji}X_j(t)-\theta_i \tag{4.2.2}$$

网络的状态向量为 $\boldsymbol{X}(t)\in\{1,-1\}^n$，且 $w_{ii}=0,i=1,2,\cdots,n$.

二、网络的工作方式

1. 串行(异步)工作方式

任一时刻 t，只有某一个节点 i(随机地或确定性地选择)依据(4.2.1)和(4.2.2)式变化，而其余 $n-1$ 个节点的状态保持不变，即：

$$\begin{aligned}X_i(t+1)&=\operatorname{sgn}(H_i(t))\\X_j(t+1)&=X_j(t),\qquad\forall j\neq i\end{aligned} \tag{4.2.3}$$

2. 并行(同步)工作方式

任一时刻 t，所有的节点都依据式(4.2.1)和(4.2.2)改变状态，即：

$$X_i(t+1)=\operatorname{sgn}(\sum_{j=1}^{n}w_{ji}X_j(t)-\theta_i),\quad\forall i \tag{4.2.4}$$

三、网络的状态

若网络从一个初态 $\boldsymbol{X}(t_0)$ 出发，经过一个有限时刻 t，网络的状态不再发生变化，即：

$$\boldsymbol{X}(t_0+t+\Delta t)=\boldsymbol{X}(t_0+t),\quad\Delta t>0 \tag{4.2.5}$$

则称网络是稳定的，这时所有的节点输出不再变化，网络稳定在某一状态.

串行方式的稳定性称为网络串行稳定性，并行方式的稳定性称为网络的并行稳定性.

4.2.2 网络的稳定性定理

一个离散型 Hopfield 神经网络可能有一个稳定点，或者有多个稳定点，也可能没有稳定点，其中有些即使是渐进稳定点，只要存在一个极限环，此网络就不可避免地在某时刻会出现振荡. 对于一个离散型反馈网络，如何依据一个解析判据来分析其稳定性，是本节的研究内容.

首先，引用铁磁材料中的哈密顿函数的形式来定义一个离散型反馈网络的能量函数. 在铁磁材料中，铁磁分子的自旋只有两个方向，记为 $P_i\in\{-1,1\},i=1,2,\cdots,n$，这种材料中的哈密顿函数为：

$$H=-\frac{1}{2}\sum_i\sum_j J_{ij}P_iP_j-\sum_i H_iP_i \tag{4.2.6}$$

其中，J_{ij} 表示第 i 个分子与第 j 个分子之间的作用，且 $J_{ij}=J_{ji}$，H_i 是外加的随机场，整个物质的相互作用结果是使哈密顿函数 H 达到最小.

由于离散型 Hopfield 神经网络的状态值也是二值量，而相互作用由权值 w_{ij} 来表示，外加输入为 I_i，所以 Hopfield 定义了一个网络的能量函数：

$$E=-\frac{1}{2}\sum_i\sum_j w_{ij}X_iX_j-\sum_i I_iX_i \tag{4.2.7}$$

由于 X_i、X_j 只可能为 1 或者为 -1，w_{ij}、I_i 有界，$i,j=1,2,\cdots,n$，所以能量函数 E 也是有界的：

$$\begin{aligned}|E|&\leqslant\frac{1}{2}\sum_i\sum_j|w_{ij}||X_i||X_j|+\sum_i|I_i||X_i|=\\&\frac{1}{2}\sum_i\sum_j|w_{ij}|+\sum_i|I_i|\end{aligned} \tag{4.2.8}$$

网络从任一状态开始，若在每次迭代中都能满足 $\Delta E \leqslant 0$，那么网络的能量将会越来越小，最后趋于稳定点($E \to 0$). 其物理意义是，在那些渐进稳定点的吸引域内，当状态离吸引子越远，其能量越大，当状态接近于稳定点时，其能量越小. 而能量函数的单调下降，说明状态的迁移是从远离吸引于到逐渐接近吸引子，最后使网络达到稳定.

一、串行方式

定理 4.1 当网络工作在串行方式下，满足 $w_{ij}=w_{ji}, w_{ii}=0, i、j=1,2,\cdots,n$，则能量函数单调下降，且网络必定稳定.

证明：对于 DHNN 网络的任一个节点 i，它的输出变化可能为：

$$\Delta X_i = X_i(t+1) - X_i(t) = \begin{cases} 0 & X_i(t+1) = X_i(t) \\ +2 & X_i(t+1) = 1, X_i(t) = -1 \\ -2 & X_i(t+1) = -1, X_i(t) = 1 \end{cases} \tag{4.2.9}$$

根据串行方式的定义，每个时刻只有一个节点发生变化，若第 i 个节点变化，而其他的节点不变，根据公式(4.2.7)可得：

$$\Delta E = -\frac{1}{2}\sum_{j=1}^{n} w_{ij} X_j(t) \Delta X_i - \frac{1}{2}\sum_{j=1}^{n} w_{ji} X_j(t) \Delta X_i - I_i \Delta X_i$$

因为

$$w_{ij} = w_{ji} \qquad w_{ii} = 0$$

所以

$$\Delta E = -(\sum_{j=1}^{n} w_{ji} X_j(t) + I_i) \Delta X_i = - H_i(t) \Delta X_i \tag{4.2.10}$$

因为

$$X_i(t+1) = \text{sgn}(H_i(t)), \quad \Delta X_i = X_i(t+1) - X_i(t)$$

因此

$$\text{当 } H_i(t) \geqslant 0 \text{ 时}, \quad \Delta X_i \geqslant 0, \quad \Delta E \leqslant 0$$
$$\text{当 } H_i(t) < 0 \text{ 时}, \quad \Delta X_i \leqslant 0, \quad \Delta E \leqslant 0$$

所以网络无论在什么条件下都能保证 $\Delta E \leqslant 0$，这样就保证了网络的稳定性和收敛性.

定理 4.2 当网络工作在串行方式下，满足 $w_{ij}=w_{ji}, w_{ii}>0, i、j=1,2,\cdots,n$，则能量函数单调下降，且网络必定稳定.

证明：对于 DHNN 网络的第 i 个节点发生变化，因为 $w_{ij}=w_{ji}$，且 $w_{ii}>0$ 则

$$\begin{aligned} \Delta E = & -\frac{1}{2}\sum_{j\neq i}^{n} w_{ij} X_j(t) \Delta X_i - \frac{1}{2}\sum_{j\neq i}^{n} w_{ji} X_j(t) \Delta X_i - \\ & w_{ii}[X_i^2(t+1) - X_i^2(t)] - I_i \Delta X_i = \\ & -(\sum_{i=1}^{n} w_{ji} X_j(t) + I_i) \Delta X_i - w_{ii} \Delta X_i X_i(t+1) = \\ & - H_i(t) \Delta X_i - w_{ii} \Delta X_i X_i(t+1) \end{aligned} \tag{4.2.11}$$

由于 $w_{ii}>0$，且 ΔX_i 与 $X_i(t+1)$ 是同号的，即能保证 $\Delta E \leqslant 0$，这样就保证了网络的稳定性和收敛性.

二、并行方式

定理 4.3 当网络工作在并行方式下，满足 $w_{ij}=w_{ji}$，则网络或者收敛于一个稳定点，或者

收敛于极限环为 2 的一个周期解.

证明:在并行工作方式时,其能量函数可以用下式表示:

$$E=-\frac{1}{2}\sum_{i}\sum_{j}w_{ij}X_i(t+1)X_j(t)-\frac{1}{2}\sum_{i}I_i(X_i(t)+X_i(t+1)) \tag{4.2.12}$$

可以写成矩阵的形式

$$E=-\frac{1}{2}\boldsymbol{X}^{\mathrm{T}}(t+1)\boldsymbol{W}\boldsymbol{X}(t)-\frac{1}{2}\boldsymbol{I}^{\mathrm{T}}[\boldsymbol{X}(t+1)+\boldsymbol{X}(t)] \tag{4.2.13}$$

$$\boldsymbol{X}\in\mathbf{R}^n;\boldsymbol{W}\in\mathbf{R}^{n\times n};\boldsymbol{I}\in\mathbf{R}^n$$

$$\begin{aligned}\Delta E=&-\frac{1}{2}\boldsymbol{X}^{\mathrm{T}}(t+1)\boldsymbol{W}\boldsymbol{X}(t)-\frac{1}{2}\boldsymbol{I}^{\mathrm{T}}[X(t+1)+\boldsymbol{X}(t)]+\\&\frac{1}{2}\boldsymbol{X}^{\mathrm{T}}(t)\boldsymbol{W}X(t-1)+\frac{1}{2}\boldsymbol{I}^{\mathrm{T}}[X(t)+\boldsymbol{X}(t-1)]=\\&-\frac{1}{2}[\boldsymbol{X}^{\mathrm{T}}(t)\boldsymbol{W}][X(t+1)-X(t-1)]-\frac{1}{2}\boldsymbol{I}^{\mathrm{T}}[\boldsymbol{X}(t+1)-\boldsymbol{X}(t-1)]=\\&-\frac{1}{2}[\boldsymbol{X}^{\mathrm{T}}(t)\boldsymbol{W}+\boldsymbol{I}^{\mathrm{T}}][X(t+1)-\boldsymbol{X}(t-1)]=\\&-\frac{1}{2}[\boldsymbol{H}(t)]^{\mathrm{T}}[X(t+1)-\boldsymbol{X}(t-1)]\end{aligned} \tag{4.2.14}$$

由于在 $\boldsymbol{H}(t)$ 中的每个分量 $H_i(t)$ 与在 $X(t+1)$ 中每个分量 $X_i(t+1)$ 同号,因而

$$[\boldsymbol{H}(t)]^{\mathrm{T}}[X(t+1)-\boldsymbol{X}(t-1)]\geqslant 0$$

所以 $\Delta E\leqslant 0$. 现在考虑在稳定点的情况,即 $\Delta E=0$ 的情况:

若 $X(t)=X(t+1)=X(t-1)$,则 $\Delta E=0$,且网络达到稳定.

若 $\boldsymbol{X}(t)\neq\boldsymbol{X}(t+1)=\boldsymbol{X}(t-1)$,则 $\Delta E=0$,且网络到达周期为 2 的极限环.

证毕.

推论:(1)如果 $\boldsymbol{W}$ 为一个正定矩阵,$I_i=0$,对所有的 i 成立,则:

$$X(t+1)=\boldsymbol{X}(t)$$

$$\boldsymbol{X}(t+1)=\mathrm{sgn}(\boldsymbol{W}\boldsymbol{X}(t))=\boldsymbol{X}(t)$$

网络必定达到稳定收敛.

(2)如果 $\boldsymbol{W}$ 为一个负定矩阵,$I_i=0$,对所有的 i 成立,则:

$$X(t+1)\neq\mathrm{sng}(\boldsymbol{W}\boldsymbol{X}(t))=-\boldsymbol{X}(t)$$

$$\boldsymbol{X}(t+1)=-\boldsymbol{X}(t)=X(t-1)$$

网络周期振荡,极限环为 2.

4.2.3 网络权值的学习

一、外积型网络权值的学习方法

外积型网络权值的学习方法使用 Hebb 规则来调整网络的权值. 网络待记忆的学习样本有 N 个,$\boldsymbol{X}^K$,$K=1,2,\cdots,N$,$X^K\in\mathbf{R}^n$,其每个分量为 X_i^K,$i=1,2,\cdots,n$,利用已知需要存储的样本来设计 n 个节点间的连接权值,如节点 i 和 j 间的连接权值为:

$$\begin{cases}w_{ij}=\alpha\sum_{K=1}^{N}X_i^KX_j^K, & i\neq j\\ w_{ij}=0, & i=j\end{cases} \tag{4.2.15}$$

其中 α 为一个正常数，初始化时 $w_{ij}=0$，当每输入一个样本时，在权值上加修正量，即 $w_{ij}=w_{ij}+\alpha X_i^K X_j^K$，当第 K 个样本 X_i^K 和 X_j^K 同时兴奋或同时抑制时，$\alpha X_i^K X_j^K>0$. 当 X_i^K 和 X_j^K 一个兴奋一个抑制时，$\alpha X_i^K X_j^K<0$. 用 Hebb 规则修正权值可以满足 $w_{ij}=w_{ji}$ 的条件，从而使得网络在串行工作方式时保证收敛，在并行工作时系统或者收敛，或者出现极限环为 2 的振荡.

把公式(4.2.15)写成矩阵的形式，取 $\alpha=1$，对于需要记忆的样本 $X^K, K=1,2,\cdots N$，权值为：

$$\boldsymbol{W}=[X^1\boldsymbol{X}^2\cdots\boldsymbol{X}^N]\begin{bmatrix}\boldsymbol{X}^{1\mathrm{T}}\\ \boldsymbol{X}^{2\mathrm{T}}\\ \vdots\\ X^{N\mathrm{T}}\end{bmatrix}-N\boldsymbol{U}=$$

$$\begin{bmatrix}X_1^1 & X_1^2 & \cdots & X_1^K & \cdots & X_1^N\\ X_2^1 & X_2^2 & \cdots & X_2^K & \cdots & X_2^N\\ \vdots & \vdots & & \vdots & & \vdots\\ X_n^1 & X_n^2 & \cdots & X_n^K & \cdots & X_n^N\end{bmatrix}\begin{bmatrix}X_1^1 & X_2^1 & \cdots & X_n^1\\ X_1^2 & X_2^2 & \cdots & X_n^2\\ \vdots & \vdots & & \vdots\\ X_1^N & X_2^N & \cdots & X_n^N\end{bmatrix}-N\begin{bmatrix}1 & 0 & \cdots & \cdots & 0\\ 0 & 1 & \ddots & & \vdots\\ \vdots & \ddots & 1 & \ddots & \vdots\\ \vdots & & \ddots & 1 & 0\\ 0 & \cdots & \cdots & 0 & 1\end{bmatrix}=$$

$$\begin{bmatrix}\sum_{K=1}^N (X_1^K)^2 & \sum_{K=1}^N X_1^K X_2^K & \cdots & \sum_{K=1}^N X_1^K X_n^K\\ \sum_{K=1}^N X_2^K X_1^K & \sum_{K=1}^N (X_2^K)^2 & \cdots & \sum_{K=1}^N X_2^K X_n^K\\ \vdots & \vdots & & \vdots\\ \sum_{K=1}^N X_n^K X_1^K & \cdots & \cdots & \sum_{K=1}^N (X_n^K)^2\end{bmatrix}-N\begin{bmatrix}1 & 0 & \cdots & \cdots & 0\\ 0 & 1 & \ddots & & \vdots\\ \vdots & \ddots & 1 & \ddots & \vdots\\ \vdots & & \ddots & 1 & 0\\ 0 & \cdots & \cdots & 0 & 1\end{bmatrix}$$

由于

$$\sum_{K=1}^N (X_1^K)^2=\sum_{K=1}^N (X_2^K)^2=\cdots=\sum_{K=1}^N (X_n^K)^2=N$$

这里，$\boldsymbol{U}$ 是一个单位矩阵. 所以

$$\boldsymbol{W}=\begin{bmatrix}0 & \sum_{K=1}^N X_1^K X_2^K & \cdots & \sum_{K=1}^N X_1^K X_n^K\\ \sum_{K=1}^N X_2^K X_1^K & 0 & \cdots & \vdots\\ \vdots & \vdots & & \vdots\\ \sum_{K=1}^N X_n^K X_1^K & X_1^K & \cdots & 0\end{bmatrix} \tag{4.2.16}$$

因此，Hebb 规则能够满足 $w_{ii}=0, w_{ij}=w_{ji}$ 的条件，网络能够收敛于稳定点.

二、稳定点的讨论

(1)如果待记忆的样本是两两正交，即对于样本 $X^K, K=1,2,\cdots,N$，X^i、X^j 为 $\boldsymbol{X}^K$ 中的任意两个不同的样本，且满足 $[X^i]^{\mathrm{T}}[\boldsymbol{X}^j]=0, i\neq j$，对所有的 i 和 j 成立.

在节点的输出为 $X_i \in \{1, -1\}$ 的情况下，当二个 n 维样本向量的各个分量中有 $\frac{n}{2}$ 个是相同的，另 $\frac{n}{2}$ 个是相反的，就能满足这两个向量正交. 用上面的外积法所得到的权值进行迭代计算，在输入样本中任取一个样本 $\boldsymbol{X}^K$ 作为初始输入，可得：

$$\boldsymbol{W}\boldsymbol{X}^K = [\boldsymbol{X}^1 \boldsymbol{X}^2 \cdots \boldsymbol{X}^K \cdots \boldsymbol{X}^N] \begin{bmatrix} \boldsymbol{X}^{1T} \\ \boldsymbol{X}^{2T} \\ \vdots \\ \boldsymbol{X}^{KT} \\ \vdots \\ \boldsymbol{X}^{NT} \end{bmatrix} \boldsymbol{X}^K - N\boldsymbol{U}\boldsymbol{X}^K =$$

$$[\boldsymbol{X}^1 \boldsymbol{X}^2 \cdots \boldsymbol{X}^K \cdots \boldsymbol{X}^N] \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \boldsymbol{X}^{KT}\boldsymbol{X}^K \\ \vdots \\ 0 \end{bmatrix} - N\boldsymbol{U}\boldsymbol{X}^K =$$

$$n\boldsymbol{X}^K - N\boldsymbol{X}^K = (n-N)\boldsymbol{X}^K \tag{4.2.17}$$

只要满足 $n > N$，则 $\mathrm{sgn}[\boldsymbol{W}\boldsymbol{X}^K] = \boldsymbol{X}^K$，则 $\boldsymbol{X}^K$ 为网络的一个稳定点.

(2)如果 N 个样本 $\boldsymbol{X}^K, K=1,2,\cdots,N$，不是两两正交，其连接权值依据 Hebb 规则求得，在 N 个样本中任选一个样本 $\boldsymbol{X}^K$ 作为初始输入：

$$\boldsymbol{W}\boldsymbol{X}^K = [\boldsymbol{X}^2 \boldsymbol{X}^2 \cdots \boldsymbol{X}^K \cdots \boldsymbol{X}^N] \begin{bmatrix} \boldsymbol{X}^{1T} \\ \boldsymbol{X}^{2T} \\ \vdots \\ \boldsymbol{X}^{KT} \\ \vdots \\ \boldsymbol{X}^{NT} \end{bmatrix} \boldsymbol{X}^K - N\boldsymbol{X}^K$$

通过上式可求得新的输出 $\boldsymbol{X}^{K'} = \mathrm{sgn}(\boldsymbol{W}\boldsymbol{X}^K)$，取 $\boldsymbol{X}^{K'}$ 的第 j 个分量：

$$\begin{aligned} X_j^{K'} &= \mathrm{sng}[\sum_{i=1}^{n} w_{ij} X_j^K] = \\ &\mathrm{sgn}[\sum_{i=1}^{n} \boldsymbol{X}_i^1 X_j^1 X_i^K + \sum_{i=1}^{n} X_i^2 X_j^2 X_i^K + \cdots + \sum_{i=1}^{n} \boldsymbol{X}_i^N \boldsymbol{X}_j^N X_i^K] = \\ &\mathrm{sgn}[\sum_{i=1}^{n} X_j^K (\boldsymbol{X}_i^K)^2 + \sum_{k=1, k\neq K}^{N} \sum_{i=1}^{n} X_i^k \boldsymbol{X}_j^k \boldsymbol{X}_i^K] = \\ &\mathrm{sgn}(s_j + n_j) \end{aligned} \tag{4.2.18}$$

式中

$$s_j = nX_j^K, \qquad n_j = \sum_{k=1, k\neq K}^{N} \sum_{i=1}^{n} X_i^k \boldsymbol{X}_j^k X_i^K$$

设 n_j 为零均值的随机变量，$X_i^k, \boldsymbol{X}_j^k, X_i^k \in \{1, -1\}$，而 n_j 的方差

$$\sigma^2 = (N-1)n, \qquad \sigma = \sqrt{(N-1)n}$$

对于非正交的学习样本，如果满足 $n > \sqrt{(N-1)n}$，则网络仍可收敛到其记忆样本上.

从上面的分析可以看出，对于正交样本，按照 Hebb 学习规则，它的稳定点与要记忆的样本相同，对于非正交的记忆样本，网络不能保证收敛到所希望的记忆样本上，只有当 $s_j > n_j$ 时，其输出近似为所要求的输出.

4.2.4 网络的稳定性实验

假定网络由 n 个节点构成，在时刻 t，网络的状态可用全部节点的输出值组成的 n 维向量

$$\boldsymbol{X}(t) = [X_1(t), X_2(t), \cdots, X_n(t)]$$

来表示，且 $\boldsymbol{X}_i(t) \in \{-1, 1\}$. 假定网络是串行工作方式，节点间是对称结合的权值矩阵. 在时刻 t，任意节点 i 的输入信号加权和为：

$$H_i(T) = \sum_{j=1}^{n} W_{ji} \cdot X_j(t) - \theta_i$$

网络状态变化，遵循下面规则：

①从网络 n 个节点中随机地选取节点 i；

②计算节点 i 的 $H_i(t)$ 值；

③根据 $H_i(t)$ 值更新节点 i 的输出 $X_i(t+1)$，

$$\begin{aligned} \text{if} \quad & H_i(t) \geqslant 0 \\ & X_i(t+1) = 1 \\ \text{Else} \quad & X_i(t+1) = -1 \end{aligned}$$

④i 以外的节点 $j(j \neq i)$ 输出不变化，

$$X_j(t+1) = X_j(t)$$

⑤返回①.

网络状态变化和能量之间的关系如图 4.2 所示，可见，遵循状态变化规则，网络的状态总是朝能量根小点方向变化，一旦落入某个根小点，网络的状态就不再变化.

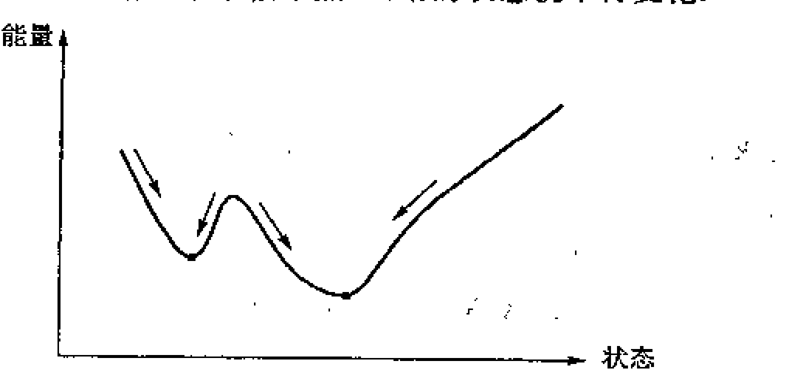


图 4.2 网络状态变化和网络能量间的关系

图 4.3 给出一个模拟有四个节点的网络状态变化的结果，分别以 16 个状态为初始状态，按照网络状态变化规则不断变化. 图(a)中的状态 b 的能量为 294，其变化为：b→f→d→5→4，最终收敛到能量最小的状态 4. 由图可见，无论从何种初始状态出发都是朝着能量减小的方向变化. Global 表示能量的全局极小值，Local 表示能量的局部极小值. 图(b)和图(a)相比只是所选取的随机数种类不同(分别为 8 和 7)，并且从任意初始状态出发都会收敛到能量最小的状态 e.

State	Energy		Time-Evolution	
(0	0)	···>	0044444444444444444444444444444	Global
(1	17)	···>	5544444444444444444444444444444	Global
(2	122)	···>	0004444444444444444444444444444	Global
(3	17)	···>	3333333333333333333333333333333	Local
(4	-115)	···>	4444444444444444444444444444444	Global
(5	10)	···>	5444444444444444444444444444444	Global
(6	77)	···>	6666444444444444444444444444444	Global
(7	80)	···>	5554444444444444444444444444444	Global
(8	122)	···>	8000444444444444444444444444444	Global
(9	187)	···>	99dd555554444444444444444444444	Global
(a	351)	···>	b333333333333333333333333333333	Local
(b	294)	···>	ffffdd54444444444444444444444	Global
(c	-66)	···>	cc44444444444444444444444444444	Global
(d	107)	···>	cc44444444444444444444444444444	Global
(e	233)	···>	cccccc4444444444444444444444444	Global
(f	284)	···>	ffdd544444444444444444444444444	Global

(a)

State	Energy		Time-Evolution	
(0	0)	···>	8cccccccccccccccccccccccccccccc	Global
(1	-72)	···>	9dddddccccccccccccccccccccccccc	Global
(2	-93)	···>	6445ddddccccccccccccccccccccccc	Global
(3	114)	···>	155ddcccccccccccccccccccccccccc	Global
(4	-115)	···>	4cccccccccccccccccccccccccccccc	Global
(5	-137)	···>	5555555555ddccccccccccccccccccc	Global
(6	-39)	···>	45dcccccccccccccccccccccccccccc	Global
(7	32)	···>	7fdcccccccccccccccccccccccccccc	Global
(8	-89)	···>	9ddddddddddcccccccccccccccccccc	Global
(9	-116)	···>	Dcccccccccccccccccccccccccccccc	Global
(a	-87)	···>	Acccccccccccccccccccccccccccccc	Global
(b	-21)	···>	999999ddccccccccccccccccccccccc	Global
(c	-233)	···>	ccccccccccccccccccccccccccccccc	Global
(d	-210)	···>	dddcccccccccccccccccccccccccccc	Global
(e	-248)	···>	ccccccccccccccccccccccccccccccc	Global
(f	-132)	···>	fffcccccccccccccccccccccccccccc	Global

(b)

图 4.3　网络状态变化的模拟结果

4.2.5　联想记忆

一、联想记忆的原理

联想可以理解为从一种事物联系到与其相关事物的过程. 由于反馈网络会收敛于其稳定状态(吸引子),使得它可以作为联想存储器(Content Addressable Memory). 具体讲,就是合理地选取权系数,使得网络的稳态恰好为联想存储的一组状态 N. 如果网络的初态在 N 中,则网络的状态将不变;如果不在 N 中,希望网络所达到的稳定状态应为 N 中与初态 Hamming 距离最近的稳定状态.

联想记忆有自联想和他联想两种形式. 自联想指的是由某种事物的全局特征或局部特征联想到其所表示的实际事物,他联想指的是由某种事物的全局特征或局部特征联想到与其密切相关的事物.

(1)自联想记忆(Auto-AM). 设在学习过程中存入 N 个样本 $\boldsymbol{X}^K, K=1,2,\cdots,N$,若输入 $\boldsymbol{X}'=\boldsymbol{X}^K+\boldsymbol{V}$,其中 $\boldsymbol{X}^K$是 N 个样本之一,$\boldsymbol{V}$ 是偏差项(可能是噪声、图形的缺损或畸变等),要求输出为 $\boldsymbol{Y}=\boldsymbol{X}^K$,即使之复原.

(2)他联想记忆(Hetero-AM). 规定两组样本之间有一定的对应关系 $\boldsymbol{X}^K\to\boldsymbol{Y}^K$, $K=1,2,\cdots,N$,例如,$\boldsymbol{X}^K$代表某人的照片,$\boldsymbol{Y}^K$代表某人的姓名. 使用时,若输入 $\boldsymbol{X}'=\boldsymbol{X}^K+\boldsymbol{V}$,要求输出为 $\boldsymbol{Y}=\boldsymbol{Y}^K$.

人脑对给出的一种事物得出与其对应事物的途径有两种:一种是按时间顺序对相关事物进行思考,例如,通过时间安排表来回忆某一段工作. 另一种就是通过事物本质特性的对比来确认事物的属性,从提示信息或局部信息对事物进行回忆或确认. 这两种途径抽象成计算机中按地址寻址和按内容寻址两种方法. 按内容寻址是基于事物全部或部分特征来找出目标事物,寻找过程就是特征对比,而不必知道这些事物的具体存储地址. 从匹配过程来看,这种方法不需要地址的管理及变换. 从观念上来看,这种方法更接近人类的思维方法,因为在人脑辨识决策过程中,绝大多数是基于事物之间的联系,也就是联想过程.

二、联想记忆的学习

为使得网络具有联想记忆功能的关键问题是如何通过学习找到合适的权值矩阵. 对于 Hopfield 网络可以用 Hebb 学习规则进行学习训练. 设网络有 n 个节点,且每个节点的输出只能取 0 或者 1,分别表示抑制和兴奋状态,学习过程中 W_{ij} 的调整原则是,若 i 和 j 两个节点同时处于兴奋状态,那么它们之间的连接应加强,即:

$$\Delta w_{ij} = \alpha X_i^K X_j^K, \qquad \alpha > 0 \tag{4.2.19}$$

具体的实现是使用外积规则. 对于给定的一组输入样本 $\boldsymbol{X}^K, K=1,2,\cdots,N$,外积规则可以表示为:

$$\boldsymbol{W} = \sum_{K=1}^{N}(\boldsymbol{X}^K[\boldsymbol{X}^K]^{\mathrm{T}} - \boldsymbol{I}),\boldsymbol{I}\text{ 是单位矩阵} \tag{4.2.20}$$

可以用下面步骤完成:

①置 $\boldsymbol{W}=[0]$;

②输入 $\boldsymbol{X}^K, K=1,2,\cdots,N$,对所有的连接对($i$ 和 j)令:

$$W_{ij} = W_{ij} + \alpha X_i^K X_j^K \tag{4.2.21}$$

下面讨论上述方法的合理性. 假定各个输入样本是正交的,且 $n>N$.

$$\begin{aligned}\boldsymbol{W}\boldsymbol{X}^1 &= (\boldsymbol{X}^1[\boldsymbol{X}^1]^{\mathrm{T}} - \boldsymbol{I}) \cdot \boldsymbol{X}^1 + \sum_{K=2}^{N}(\boldsymbol{X}^K[\boldsymbol{X}^K]^{\mathrm{T}} - \boldsymbol{I}) \cdot \boldsymbol{X}^1 = \\ &(n-1) \cdot \boldsymbol{X}^1 - (N-1) \cdot \boldsymbol{X}^1 = \\ &(n-N) \cdot \boldsymbol{X}^1\end{aligned} \tag{4.2.22}$$

所以 $\mathrm{sgn}(\boldsymbol{W}\boldsymbol{X}^1)=\boldsymbol{X}^1$,可见 $\boldsymbol{X}^1$是一个稳定状态.

三、联想记忆的模拟实现

图 4.4 表示利用联想记忆功能实现图像的联想. 设网络的节点以二维矩阵的形式排列,节点输出为 1 时,用小圆圈表示,输出为 0 时,用小点表示. 并且网络的一个稳态,即网络能量函数的极小点,对应一幅完整的图像. 其周围状态对应不完整的图像(缺损图像). 那么,从不完整的图像对应的状态为初始状态出发,经过状态的不断迁移,可以收敛到完整图像所对应的状态. 即可以理解为从不完整的图像联想起完整图像的过程.

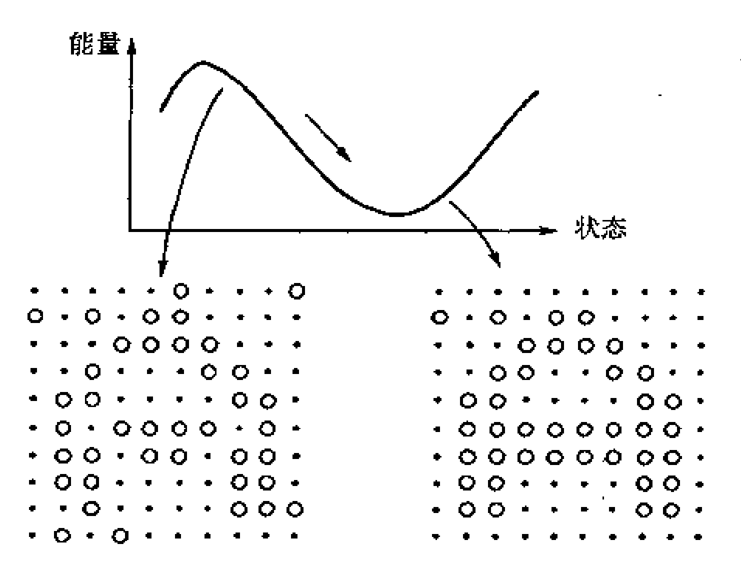


图 4.4　联想记忆的过程

假定所希望的记忆模式为 N 个，即对应网络 N 个能量极小点，对于节点 i 和节点 j 可以用下式确定网络的连接权值；

$$w_{ij} = \sum_{K=1}^{N}(2X_i^K - 1)(2X_j^K - 1) \tag{4.2.23}$$

$$w_{ii} = 0, \theta_i = 0$$

且记忆模式满足下面的约束条件：

$$2\sum_{j=1}^{N} X_j^P X_j^Q - \sum_{j=1}^{n} X_j^Q \approx 0 \tag{4.2.24}$$

即从 N 个模式中任取两个模式 P 和 Q，两个模式向量的分量同时为"1"的数为每个模式向量中"1"的个数的一半．倘若记忆模式满足式(4.2.24)的制约条件，且按式(4.2.23)确定网络权值，则待记忆的模式对应的网络状态都是网络的稳态，即对应网络能量的极小点．另外，不仅记忆模式本身，而且各记忆模式向量的各分量求反后所得的模式也成为网络能量的极小点，由下式可以证明之：

$$w_{ij} = \sum_{K=1}^{N}[2(1 - X_i^K) - 1][2(1 - X_j^K) - 1] = \sum_{K=1}^{N}(2X_i^K - 1)(2X_j^K - 1) \tag{4.2.25}$$

模拟实现联想记忆的记忆模式如图 4.5 所示．网络的节点数为 10×10=100 个，图 4.6 为模拟实验的输出结果，其中 D 表示海明距离，E 表示网络所对应的能量，Rev 表示网络状态的

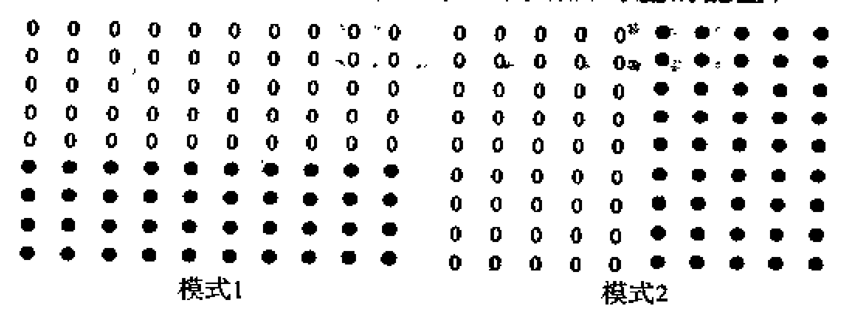


图 4.5　输入模式

各分量求反后所对应的网络状态，图 4.6 右面的矩阵中的数表示各节点使用规则更新的次数.从图中可以看出，从初态开始，能量为 4，与模式 1 的海明距离为 40，与模式 2 的海明距离为 50，经 200 次状态变化，最终收敛到模式 1 的状态.

[初始状态] E=4	0	●	●	0	●	0	●	●	●	●	0	0	0	0	0	0	0	0	0	0
	0	●	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	●	●	0	0	●	0	0	0	●	0	0	0	0	0	0	0	0	0	0
	0	0	0	●	0	0	0	●	0	0	0	0	0	0	0	0	0	0	0	0
	●	0	0	●	●	0	0	0	●	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	●	●	●	●	0	0	●	0	0	0	0	0	0	0	0	0	0
	●	0	●	0	●	●	●	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	●	●	0	●	●	●	0	●	0	0	0	0	0	0	0	0	0	0
	0	0	●	●	0	●	●	●	●	0	0	0	0	0	0	0	0	0	0	0
	●	0	●	●	●	●	0	0	●	0	0	0	0	0	0	0	0	0	0	0

Pat	D	E	Rev	D	E
[0]:	40	−1200	([0]:	60	−1200)
[1]:	50	−1200	([1]:	50	−1200)

[100次后] E=−628	0	0	0	0	●	0	●	0	0	0	1	1	1	1	0	0	0	1	1	1
	●	0	0	0	0	0	0	0	●	0	1	2	1	1	0	0	1	1	1	1
	0	●	0	0	0	●	0	0	0	0	0	1	3	0	1	3	1	0	2	1
	0	0	0	●	0	0	●	0	0	0	2	1	2	0	1	0	1	2	1	0
	0	0	0	0	0	0	0	0	0	●	1	1	2	1	1	1	2	2	1	1
	0	0	0	●	●	●	●	●	0	●	0	0	0	0	1	1	1	3	0	1
	●	0	●	●	●	●	●	0	●	●	0	0	1	4	2	0	0	0	0	1
	●	●	●	●	●	●	●	●	0	●	2	1	2	0	2	0	0	1	0	2
	0	●	●	●	●	●	●	●	●	0	0	4	0	0	2	2	1	0	3	0
	●	●	●	●	●	●	●	●	●	●	1	1	2	0	1	2	1	1	0	4

Pat	D	E	Rev	D	E
[0]:	13	−1200	([0]:	87	−1200)
[1]:	51	−1200	([1]:	49	−1200)

[200次后] E=−1200	0	0	0	0	0	0	0	0	0	0	4	3	5	1	3	5	3	1	3	3
	0	0	0	0	0	0	0	0	0	0	2	5	4	4	1	1	3	4	6	1
	0	0	0	0	0	0	0	0	0	0	1	2	7	2	5	4	5	2	3	3
	0	0	0	0	0	0	0	0	0	0	2	1	3	1	4	2	4	5	2	1
	0	0	0	0	0	0	0	0	0	0	5	3	3	4	2	3	2	3	3	3
	●	●	●	●	●	●	●	●	●	●	3	1	1	1	6	1	2	4	3	1
	●	●	●	●	●	●	●	●	●	●	2	5	4	6	3	2	2	3	3	5
	●	●	●	●	●	●	●	●	●	●	3	2	4	2	3	0	2	2	3	3
	●	●	●	●	●	●	●	●	●	●	1	5	2	3	3	5	7	2	5	2
	●	●	●	●	●	●	●	●	●	●	3	3	5	1	3	5	2	2	2	6

Pat	D	E	Rev	D	E
[0]:	0	−1200	([0]:	100	−1200)
[1]:	50	−1200	([1]:	50	−1200)

图 4.6　模拟实验的输出结果

§4.3 连续型 Hopfield 神经网络

连续型 Hopfield 神经网络 CHNN 的每个节点的输入与输出关系为连续可微的单调上升函数，每个节点的输入是一个随时间变化的状态变量，它与外界输入和其他节点来的信号有直接关系，同时与其他节点之间的连接权值相关. 由于状态变量直接影响输入变量，使得系统为一个随时间变化的动态系统.

4.3.1 网络结构和数学模型

连续型 Hopfield 神经网络在结构上和离散型相同，而且状态方程形式上也相同. 若定义网络中第 i 个节点的输入为 u_i，输出为 v_i，那么输入输出的关系为：

$$v_i = f(u_i) = f\left(\sum_{j=1}^{n} w_{ji} v_j - \theta_i\right) \tag{4.3.1}$$

其中，n 为网络的节点数，状态转移函数为 Sigmoid 型函数，一般常用

$$f(x) = 1/(1+e^{-\lambda x}) \text{ 或者 } f(x) = \tanh(\lambda x)$$

网络工作方式分为异步、同步和连续更新三种方式，与离散型 Hopfield 网络相比，多了一种连续更新方式，即网络中所有节点的输出都随时间连续变化.

图 4.7 表示网络中的单个电子神经元，神经元可由一个有正负输出的运算放大器模拟，放大器的输入部分表示生物神经元的树突，输入电阻表示各个神经元之间的连接强度，放大器的输出部分代表生物神经元的轴突，电容 C_i 表示神经元的细胞膜电容，电阻 R_i 代表细胞膜电阻. 图 4.8 所示的是一种连续型 Hopfield 网络结构，该网络模型可与电子电路直接对应. 图中神经元 i 和神经元 j 之间通过电导 T_{ij} 相连，实际上，这种连接是通过电阻 $R_{ij}=1/T_{ij}$ 来实现的. 当 $T_{ij}>0$ 时，表示这种连接是兴奋型的，电阻 R_{ij} 接到放大器的正输出端上，$T_{ij}<0$ 时，表示这种连接是抑制型的，电阻 R_{ij} 接到放大器的负输出端上.

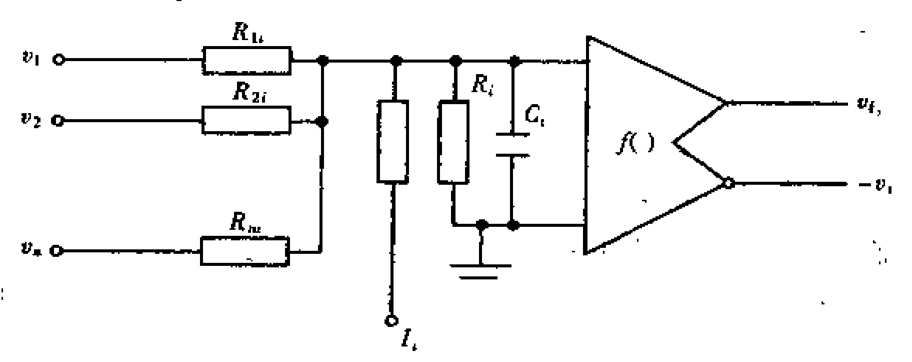


图 4.7　利用运算放大器实现的神经元

对图 4.7 的神经元，根据克希荷夫定律可写出下列微分方程：

$$\begin{aligned} C_i \frac{\mathrm{d}u_i}{\mathrm{d}t} &= \sum_{j=1}^{n} T_{ji} v_j - \frac{u_i}{R_i} + I_i \\ v_i &= f(u_i) \end{aligned} \tag{4.3.2}$$

其中，u_i，v_i 分别为神经元 i 的输入和输出电压；R_i，C_i 分别表示神经元 i 的输入电阻和输入电容；I_i 是偏量电流. $C_i \frac{\mathrm{d}u_i}{\mathrm{d}t}$ 表示输入电流流向神经元 i 的输入电容 C_i 充电至电位 u_i 的电流，这一

电流由方程右端的三部分组成. $\sum_{j=1}^{n} T_{ij}v_j$ 表示神经网络中,神经元 j 输出电压通过神经元间的联接电导 T_{ij} 产生的电流,因此这一部分是所有与神经元 i 相连接的神经元对神经元 i 所提供的电流;$\frac{u_i}{R_i}$表示神经元 i 的膜电阻的漏电流;I_i表示外部的输入电流. 不难看出,图 4.8 网络的数学模型可以用 n 个式(4.3.2)所表示的非线形方程组来描述.

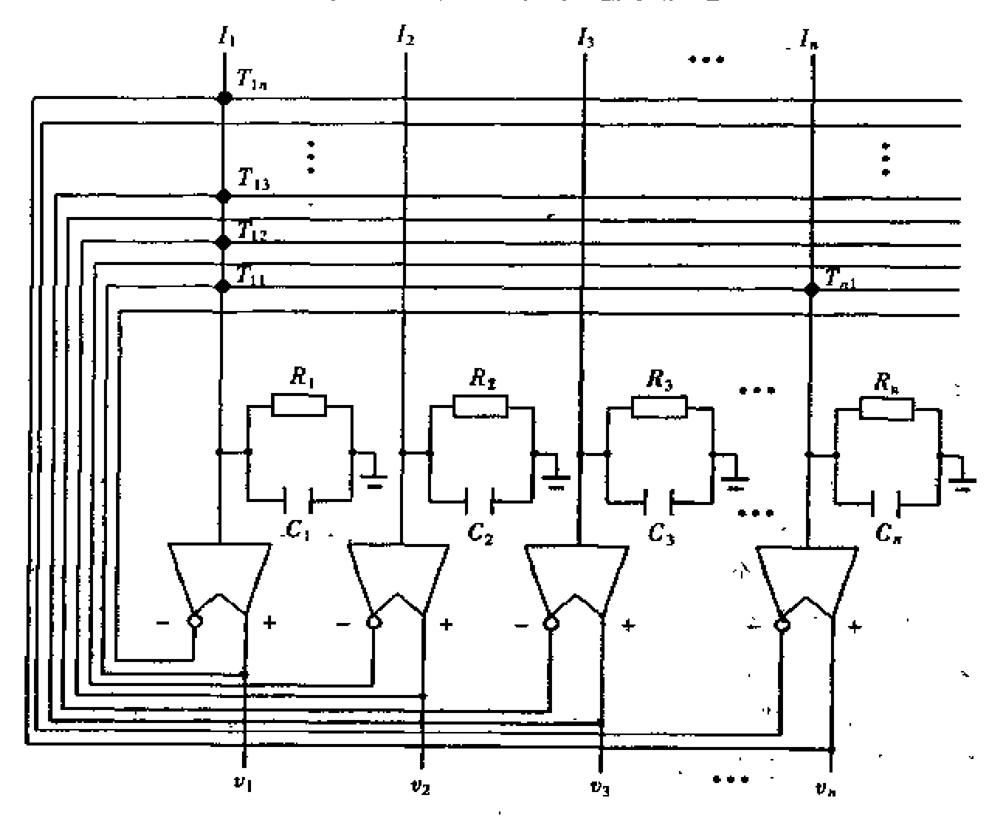


图 4.8 连续型 Hopfield 网络结构

4.3.2 网络的稳定性分析

Hopfield 将图 4.8 网络的能量函数 E 定义为:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} T_{ij}v_iv_j - \sum_{i=1}^{n} v_iI_i + \sum_{i=1}^{n} \frac{1}{R_i}\int_0^{v_i} f^{-1}(v)\mathrm{d}v \tag{4.3.3}$$

定理 4.4 假定神经元转移特性函数 $f(\cdot)$,存在反函数 $f^{-1}(\cdot)$,并且是单调连续递增函数,同时网络结构对称,即 $T_{ij}=T_{ji}$, $T_{ii}=0$,那么沿系统的运行轨迹有 $\mathrm{d}E/\mathrm{d}t\leqslant 0$,当且仅当 $\mathrm{d}v_i/\mathrm{d}t=0$ 时,$\mathrm{d}E/\mathrm{d}t=0$, $i,j\in\{1,2,\cdots,n\}$.

证明

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \sum_j \frac{\partial E}{\partial v_j}\frac{\partial u_j}{\partial t} \tag{4.3.4}$$

由式(4.3.3)和 $u_j=f^{-1}(v_j)$,对某个特定的 j 有:

$$\frac{\partial E}{\partial v_j} = -\frac{1}{2}\sum_i (T_{ji}-T_{ij})v_i - \left(\sum_i T_{ij}v_j - \frac{u_j}{R_j} + I_j\right) \tag{4.3.5}$$

由于网络的对称性和式(4.3.2),那么式(4.3.5)可写为:

$$\frac{\partial E}{\partial v_j}=-C_j\frac{\mathrm{d}u_j}{\mathrm{d}t}=-C_j(f^{-1})'\frac{\mathrm{d}v_j}{\mathrm{d}t} \tag{4.3.6}$$

将式(4.3.6)代入式(4.3.4)得

$$\frac{\mathrm{d}E}{\mathrm{d}t}=-\sum_{j\neq i}C_j(f^{-1})'\left(\frac{\mathrm{d}v_j}{\mathrm{d}t}\right)^2 \tag{4.3.7}$$

因为 $f^{-1}(\cdot)$单调递增和 $C_j>0$,因此

$$\frac{\mathrm{d}E}{\mathrm{d}t}\leqslant 0$$

只有对于所有的 j 满足$\frac{\mathrm{d}v_j}{\mathrm{d}t}$时,才有$\frac{\mathrm{d}E}{\mathrm{d}t}=0$.

证毕.

由上面定理可知,随着时间的增长,网络状态不断变化,网络能量逐渐降低.当且仅当网络中所有节点的状态不再变化时,能量也不再变化,此时到达能量的极小点.这就是说网络的稳定点就是 E 的极小点.

如果网络的结构不是对称的,则无法保证网络的稳定性.这时,在网络的运行过程中可能出现极限环或者混沌现象.

下面讨论 T_{ij} 和 w_{ij} 的关系,若式(4.3.2)不考虑外加电流 I_i,并将等式两边同乘以 R_j,那么有:

$$R_iC_i\frac{\mathrm{d}u_i}{\mathrm{d}t}=\sum_j T_{ij}R_iv_j-u_i \tag{4.3.8}$$

令 $\tau_i=R_iC_i,w_{ij}=T_{ij}R_i$,那么式(4.3.8)变为:

$$\tau_j\frac{\mathrm{d}u_i}{\mathrm{d}t}=\sum_j w_{ij}v_j-u_i \tag{4.3.9}$$

τ_i为模拟神经元的输出时间特性,在网络稳定后,由(4.3.9)式可得:

$$u_i=\sum_j w_{ji}v_j \tag{4.3.10}$$

这与式(4.3.1)的定义相一致,只不过阈值 $\theta_i=0$.

当 Hopfield 网络的节点间连接呈对称性时,即 $T_{ij}=T_{ji}$,系统的能量 E 在达到局部极小值后就停止随时间变化.这说明 Hopfield 网络构成的一个非线形动力学系统可以稳定,并且稳定在局部极值或全局极值点上.在构造一个 Hopfield 神经网络时,由于网络结构已由 Hopfield 提出,因此应用者要做的工作是将一个应用问题化成 Hopfield 网络结构所能表达的问题,同时要找到合理的能量函数表达式,然后找到相应的 T_{ij} 值.系统的稳定性问题是连续型 Hopfield 网络实理的关键问题,不然会引起网络的不稳定或者振荡,这样就达不到预期的目的.

下面对网络的能量公式给予讨论.一种异步工作方式的网络能量公式由下式表示:

$$E=-\frac{1}{2}\sum_i\sum_j T_{ij}v_iv_j-\sum_i v_iI_i+\sum_i u_iv_i \tag{4.3.11}$$

任选一个节点 i,且只有 i 的状态变化,则能量的变化量为:

$$\Delta E=-\left(\sum_{i\neq j}T_{ij}v_j+I_i-u_i\right)\Delta v_i \tag{4.3.12}$$

当 $\sum_{i\neq j}T_{ij}v_j+I_i-u_i>0$,并且 $\Delta v_i>0$ 时,$\Delta E<0$,系统可趋于稳定;

当 $\sum_{i\neq j} T_{ij}v_j + I_i - u_i < 0$，并且 $\Delta v_i<0$ 时，$\Delta E<0$，系统可趋于稳定.

对于节点 i 有：$v_i=f(u_i)$和 $u_i=f^{-1}(v_i)$，为了方便起见，假定转移函数为线性函数，即

$$v_i = f(u_i) = ku_i$$

则有 $u_i=f^{-1}(v_i)=\frac{v_i}{k}$，当 $k\to\infty$，也即神经元增益趋向无穷时，$u_i=\frac{v_i}{k}\to 0$，所以在能量公式(4.3.11)中可略去 $\sum_i u_iv_i$ 项，这时能量公式为：

$$E=-\frac{1}{2}\sum_i\sum_j T_{ij}v_iv_j - \sum_i v_iI_i \tag{4.3.13}$$

当外部流入神经网络的电流 I_i为 0 时，能量函数还可化简为：

$$E=-\frac{1}{2}\sum_i\sum_j T_{ij}v_iv_j \tag{4.3.14}$$

§4.4 Hopfield 网络的应用实例

4.4.1 用于求解 TSP 问题

TSP 问题亦称邮递员路径问题，其内容为：假定有 n 个城市 A、B、C、…，已知其相互间距离为 d_{AB}，d_{BC}，d_{CA}，…，问题是寻找一条合理的闭合路径，此路径经过每个城市，且仅经过一次，并返回到起始城市，要求总的路径：$d=d_{AB}+d_{BC}+d_{CA}+\cdots$为最短.

在给定 n 的条件下，闭合路径的数目为 $k=(n-1)!/2$，其算式为：

$$D_d = \sum_{i=1}^{n-1} d_{i,i+1} + d_{1,n}$$

$$D = \min_{d=1}^{k}(D_d)$$

用神经网络求解，首先要找到一个合适的表示方法. 以五个城市为例，可用一个矩阵来表示，如图 4.9 所示，设 C 为出发点，它表示路径的顺序为 $C\to A\to E\to B\to D\to C$. 在此矩阵中各行各列只能有一个元素为 1，其余都是零，否则它就是一个无效路径. 如果用下标 x、y 表示城市，i 表示第几次访问，即 x、$y\in\{A,B,C,D,E\}$，$i\in\{1,2,3,4,5\}$，图中表示了一条有效路径，其路径的长度为：

$$D = d_{CA} + d_{AE} + d_{EB} + d_{BD} + d_{DC}$$

	1	2	3	4	5
A	0	1	0	0	0
B	0	0	0	1	0
C	1	0	0	0	0
D	0	0	0	0	1
E	0	0	1	0	0

图 4.9 用矩阵表示 TSP 问题的访问顺序

为了解决 TSP 问题，必须构造一个有 $n\times n$ 个节点的网络，在网络运行时，网络的能量不断降低，网络稳定后其输出状态代表城市的访问顺序，即构成如图 4.9 的矩阵. 网络能量的极

小点对应最佳或准最佳路径.

解决问题的关键是找到合适的能量函数.

(1)针对有效路径,构造行约束条件:

$$E_1 = \frac{A}{2}\sum_x \sum_i \sum_{j\neq i} v_{xi} v_{xj}, \quad v_{xi} \in \{0,1\}$$

其中 $A>0$ 为常数,E_1保证当矩阵的每一行不多于一个 1 时,其值达到最小 $E_1=0$.

(2)构造列约束条件:

$$E_2 = \frac{B}{2}\sum_i \sum_x \sum_{y\neq x} v_{xi} v_{yi}$$

其中 $B>0$ 为常数,E_2保证当矩阵的每一列不多于一个 1 时,其值达到最小 $E_2=0$.

(3)构造全局约束条件:

$$E_3 = \frac{C}{2}\Big(\sum_x \sum_i v_{xi} - n\Big)^2$$

其中 $C>0$ 为常数,E_3保证当矩阵中 1 的个数恰好为 n 时,E_3达到最小 $E_3=0$.

(4)定义路径的长度为:

$$E_4 = \frac{D}{2}\sum_x \sum_{y\neq x} \sum_i d_{xy} v_{xi} (v_{y,i+1} + v_{y,i-1})$$

其中 d_{xy}表示城市 x 到城市 y 的距离,v_{xi}表示矩阵中第 x 行第 i 列的元素,其值为 1 时,表示第 i 步访问城市 x,为 0 时,表示第 i 步不访问城市 x. E_4取极小值时对应最佳路径.

因此,网络的能量函数为:

$$E = E_1 + E_2 + E_3 + E_4 \tag{4.4.1}$$

将式(4.4.1)和标准能量函效公式(4.3.13)相比可得到节点 i 和节点 j 之间的导纳值为:

$$T_{xi,yj} = A\delta_{x,y}(1-\delta_{i,j}) - B\delta_{i,j}(1-\delta_{x,y}) - C - Dd_{xy}(\delta_{j,i+1} + \delta_{j,i-1}) \tag{4.4.2}$$

其中 $\delta_{i,j}$和 $\delta_{x,y}$定义为:

$$\delta_{i,j} = \begin{cases} 1, i=j \\ 0, \text{其他} \end{cases} \quad \text{偏流为}: I_{xi} = C_n \tag{4.4.3}$$

将式(4.4.2)和(4.4.3)代入网络状态方程(4.3.2)中,得到:

$$C_{xi}\frac{\mathrm{d}u_{xi}}{\mathrm{d}t} = -\frac{u_{xi}}{R_{xi}} - A\sum_{j\neq i} v_{xi} - B\sum_{y\neq x} v_{yi} - C\Big(\sum_x \sum_i v_{xi} - n\Big) - D\sum_{y\neq x} d_{xy}(v_{y,i+1} + v_{y,i-1}) \tag{4.4.4}$$

当网络节点输出为连续型时:

$$v_{xi} = f(u_{xi}) = \frac{1}{2}\left[1 + \tanh\Big(\frac{u_{xi}}{u_0}\Big)\right]$$

当网络节点输出为离散型时:

$$v_{xi} = \begin{cases} 1, & u_{xi} \geq 0 \\ 0, & u_{xi} < 0 \end{cases}$$

选择合适的 A、B、C、D、u_0和初始值 $u_{xi}=1/n$,按上式迭代直至收敛.

下面介绍一种改进的算法. 为了简化能量函数,式(4.4.1)中的第三项仅在网络输出为全零时才起作用,否则前面网项已经保证了第三项的或立,如果前两项改为:

$$\frac{A}{2}\sum_{x=1}^{n}\Big(\sum_{i=1}^{n}v_{xi}-1\Big)^2+\frac{B}{2}\sum_{i=1}^{n}\Big(\sum_{x=1}^{n}v_{xi}-1\Big)^2$$

则第三项可以省去，于是能量函数为：

$$E=\frac{A}{2}\sum_{x=1}^{n}\Big(\sum_{i=1}^{n}v_{xi}-1\Big)^2+\frac{B}{2}\sum_{i=1}^{n}\Big(\sum_{x=1}^{n}v_{xi}-1\Big)^2+\frac{D}{2}\sum_{x=1}^{n}\sum_{y\neq x}^{n}\sum_{i=1}^{n}d_{xy}v_{xi}v_{y,i+1} \quad (4.4.5)$$

由于行和列的对称性，取 $B=A$，第三项由式(4.4.1)的$(v_{y,i+1}+v_{y,i-1})$两项简化为 $v_{y,i+1}$ 一项仍能满足约束条件的要求.由

$$\frac{\mathrm{d}u_{xi}}{\mathrm{d}t}=-\frac{\partial E}{v_{xi}} \qquad (x,i=1,2,\cdots,n)$$

导出网络的动态方程：

$$\frac{\mathrm{d}u_{xi}}{\mathrm{d}t}=-A\Big(\sum_{j=1}^{n}v_{xj}-1\Big)-A\Big(\sum_{y=1}^{n}v_{yi}-1\Big)-D\sum_{y=1}^{n}d_{xy}v_{y,i+1} \qquad (4.4.6)$$

网络权值与外部输入电流为：

$$T_{xi,yj}=-A\delta_{x,y}-A\delta_{i,j}-Dd_{xy}\delta_{j,i+1}, I_{xi}=2A \qquad (4.4.7)$$

算法步骤如下：

①置 $t=0, A=1.5, D=1$；

②读入 n 个城市之间的距离 $d_{x,y}(x,y=1,2,\cdots,n)$；

③由式(4.4.7)计算权值 $T_{xi,yj}$ 和输入偏置 I_{xi}；

④赋予 $u_{xi}(t)$ 的初始值为接近于 0 的随机数$(x,i=1,2,\cdots,n)$；

⑤计算 $v_{xi}(t),(x,i=1,2,\cdots,n), v_{xi}(t)=\frac{1}{2}\Big(1+\tanh\frac{u_{xi}(t)}{u_0}\Big)$，取 $u_0=0.02$；

⑥利用动态方程计算 $\Delta u_{xi}(t),(x,i=1,2,\cdots,n)$

$$\Delta u_{xi}(t)=\sum_{y=1}^{n}\sum_{j=1}^{n}T_{xi,yj}v_{yj}+I_{xi}$$

⑦根据一阶欧拉公式计算 $u_{xi}(t+1)$，

$$u_{xi}(t+1)=u_{xi}(t)+\Delta u_{xi}\Delta t, \qquad x,i=(1,2,\cdots,n)$$

取 $\Delta t=0.5$；

⑧如果系统达到平衡状态，则终止，否则返回⑤.

4.4.2 用于求解货流问题

如图 4.10 所示，求解货流问题的描述如下：设有 m 个货源，每个货源存量为 $S_x, x=1,2,\cdots,m$，向 n 个用户供货，每个用户需要量为 $D_y, y=1,2,\cdots,n$，C_{xy} 为从 x 运到 y 单位货物所需要费用，问题是求一个供应方案 f_{xy}，使满足各用户需要且总费用最小.

用神经网络求解该问题，首先要解决如何利用各神经元状态的组合来表示数量的问题.对数的编码通常有以下几种方案：

(1)二进制法：这是最常用的方案，用 $\log_2(N+1)$ 位表示数 N，数空间与神经元空间是一一对应的.这种方法节省神经元数量，但容错性差，尤其当表示最高位的状态错误时，误差最大.

(2)简单求和法：用处于激活状态的神经元数目之和表示数，则表示数 N 最少要用 N 个神经元，数空间和神经元空间是一对多的.这种方法不经济，但容错性好.

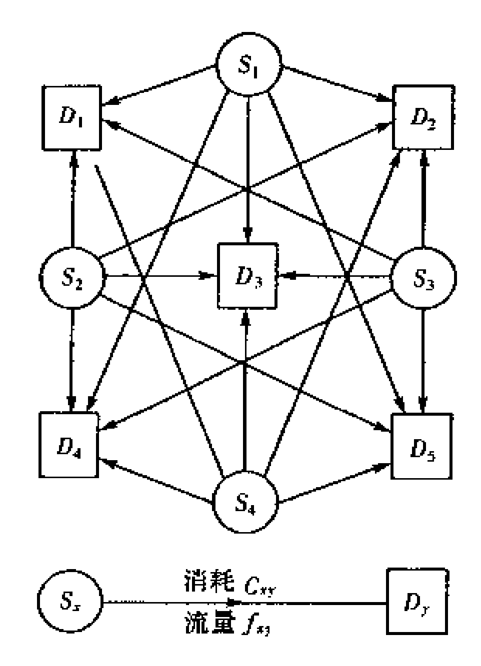


图 4.10　货流问题

(3)分组求和法:综合上两种方法,把 q 位数分成 K 组,每组 M 位($q=KM$),组内用简单求和法表示,则任何一数都可写成

$$\sum_{k=1}^{K}\left[(M+1)^{k-1}\sum_{i=1}^{M}v_{(k-1),M+i}\right]$$

例如,对 $q=6,K=2,M=3$,则 5 可表示为 100100,即 $4^1\times(1+0+0)+4^0\times(1+0+0)=5$ 或 010001,001010,100001 等.若用$\{+1,-1\}^n$表示,则在上式中加一负项:

$$\sum_{k=1}^{K}\left[(M+1)^{k-1}\sum_{i=1}^{M}v_{(k-1),M+i}\right]-\left[\frac{1}{2}(M+1)k-1\right]$$

现在来解货流问题,可以把该问题转化为在约束条件

$$\begin{cases}\sum_{y=1}^{n}f_{xy}\leqslant S_x & (x=1,2,\cdots,m)\\ \sum_{x=1}^{m}f_{xy}\leqslant D_y & (y=1,2,\cdots,n)\end{cases}$$

下求 $\sum_{x=1}^{m}\sum_{y=1}^{n}C_{xy}f_{xy}$ 的最小值.

设单位费用矩阵及存需矩阵已知,分别如表 4.1,表 4.2,流量矩阵如表 4.3,用 q 个神经元表示整个矩阵中每个元素,以示流量 f_{xy},所以总共要 $N=q_{mn}$ 个神经元表示整个流量矩阵,每个神经元状态 $v_{x,y,i}$ 有三个下标,x,y 表示神经元所属矩阵元素,i 表示它在该元素中的位置.

表 4.1　单位费用矩阵 C_{xy}

	$y=1$	$y=2$	$y=3$	$y=4$	$y=5$
$x=1$	5	1	7	3	3
$x=2$	2	3	6	9	5
$x=3$	6	4	8	1	4
$x=4$	3	2	2	2	4

表 4.2　存需矩阵

	$y=1$				$y=2$				……				$y=n$			
	i				i				……				i			
	q	…	2	1	q	…	2	1					q	…	2	1
$x=1$				$f_{1,2}$												
$x=2$			$v_{2,1,2}$													
…																
$x=m$																

表 4.3　流量矩阵

		D_1	D_2	D_3	D_4	D_5
	总量	2	7	3	2	4
S_1	5	0	5	0	0	0
S_2	3	2	1	0	0	0
S_3	4	0	0	0	2	2
S_4	6	0	1	3	0	2

采用分组求和法表示数：

$$f_{xy}=\sum_{k=1}^{K}\left[(M+1)^{k-1}\sum_{i=1}^{M}v_{xy,(k-1)M+i}\right] \tag{4.4.8}$$

目标函数为

$$\begin{aligned}I=&-\frac{A}{2}\sum_{x=1}^{m}\sum_{y=1}^{n}\sum_{k=1}^{K}\sum_{i=1}^{M}(M+1)^{k-1}[1-2v_{xy,(k-1)M+i}]^2+\\&\frac{B}{2}\sum_{x=1}^{m}\left[S_x-\sum_{y=1}^{n}\sum_{k=1}^{K}\sum_{i=1}^{M}(M+1)^{k-1}v_{xy,(k-1)M+i}\right]^2+\\&\frac{C}{2}\sum_{y=1}^{n}\left[D_y+\sum_{x=1}^{m}\sum_{k=1}^{K}\sum_{i=1}^{M}(M+1)^{k-1}v_{xy,(k-1)M+i}\right]^2+\\&\frac{D}{2}\left[\sum_{x=1}^{m}\sum_{y=1}^{n}\sum_{k=1}^{K}\sum_{i=1}^{M}C_{xy}(M+1)^{k-1}v_{xy,(k-1)M+i}\right]^2\end{aligned} \tag{4.4.9}$$

式中，A,B,C,D 皆为正的系数，第一项是使状态变量 v_{xy} 二值化，由于函数形式为

$$F(v)=-(1-2v)^2 \qquad (0\leqslant v\leqslant 1)$$

当 $v=0$ 和 $v=1$ 时达到最小. 第二项是存量约束，第三项是需量约束，第四项是使总费用最小. 网络的能量函数可表示为

$$\begin{aligned}E=&-\frac{1}{2}\sum_{x=1}^{m}\sum_{y=1}^{n}\sum_{k=1}^{K}\sum_{i=1}^{M}\sum_{x'=1}^{m}\sum_{y'=1}^{n}\sum_{k'=1}^{K}\sum_{i'=1}^{M}T_{xy,(k-1)M+i,x'y',(k'-1)M+i'}\times\\&v_{xy,(k-1)M+i}v_{x'y',(k'-1)M+i'}-\sum_{x=1}^{m}\sum_{y=1}^{n}\sum_{k=1}^{K}\sum_{i=1}^{M}v_{xy,(k-1)M+i}\times\\&I_{xy,(k-1)M+i}\end{aligned} \tag{4.4.10}$$

其中 $T_{xy,(k-1)M+i,x'y'(k'-1)M+i'}$ 表示在流量矩阵中的第 xy 元素的位于[$(k-1)M+i$]的神经元与第 $x'y'$ 元素位于[$(k'-1)M+i'$]的神经元间连线强度. 令 I 与 E 的对应项相等可得

$$T_{xy,(k-1)m+i,x'y',(k'-1)M+i'}=4A(M+1)^{k-1}\delta_{xx'}\delta_{yy'}\delta_{kk'}\delta_{ii'}-$$

$$B(M+1)^{k+k'-2}\delta_{xx'} - C(M+1)^{k+k'-2}\delta_{yy'} -$$
$$D(M+1)^{k+k'-2}C_{xy}C_{x'y'} \tag{4.4.11}$$

$$I_{xy,(k-1)M+i} = 2A(M+1)^{k-1} + B(M+1)^{k-1}S_x + C(M+1)^{k-1}D_y \tag{4.4.12}$$

其中 $\delta_{xx'}$ 为 δ 函数：

$$\delta_{xx'} = \begin{cases} 1, & z = z' \\ 0, & z \neq z' \end{cases}$$

4.4.3 在通信网络中的应用

现代通信网络是一个很复杂的系统，合理地对之进行调度和管理是通信产业的一个重要问题. 使用神经网络可以动态地解决其优化问题.

1. 路由规划(Routing)

目前的通信网络都是点到点连接的，可用一个无向图 $G=(v,e)$ 来表示，顶点代表交换节点，边代表通路，每个边有一最大容量. 为实现网络中所有点到点之间的通信，需要根据用户呼叫及网络业务量的情况合理地安排路由. 目前多用事先安排的路由表选择路由，这是一种静态方法. 由于业务量时刻变化，人们希望能据当时情况动态选路，以进一步提高效率. 神经网络的并行计算，为动态选路提供了一条可行的方案.

首先对无向图 G 的各边编号，路由经过的边用 1 表示($L_{ijk}=1$)，不经过的边用 0 表示($L_{ijk}=0$). 这样一个路由可用一个 0,1 串来表示，$L\times M$ 个神经元对应 L 个需求的 $L\times M$ 条路由，M 为备选路由数，设 G 有 N 个点，则选路的要求是：

(1)每一呼叫最多有一条路由

$$E_1 = \frac{A}{2}\sum_{i=1}^{L}\sum_{j=1}^{M}\sum_{\substack{q=1\\q\neq j}}^{N}V_{ij}V_{iq}$$

不多于一条路由时，其值达到最小 $E_1=0$.

(2)每一呼叫尽可能有一条路由，

$$E_2 = \frac{B}{2}\sum_{i=1}^{L}\left(1-\sum_{j=1}^{M}V_{ij}\right)^2$$

路由数恰好为 1 时，其值达到最小 $E_2=0$.

(3)所选路由代价(阻塞、延时)最低，

$$E_3 = \frac{C}{2}\sum_{i=1}^{L}\sum_{j=1}^{M}\sum_{p=1}^{L}V_{ij}V_{pj}\sum_{k=1}^{N}C_k(L_{ijk} \vee L_{pjk})$$

E_3 取最小值时对应最优路由.

总目标函数：

$$E = E_1 + E_2 + E_3$$

2. 排队调度(Queuing)

如果到达 ATM 交换单元的两条入线的两个以上信元在同一时刻去往同一出线，则必须排队. 用神经网络可以实现这种排队的调度问题.

设网络由 N 个神经元组成，窗口大小为 W，即每个队列的前 W 个信元将顺序竞争发送权，按如下方式确定能量函数.

(1)要求所有被选中神经元具有不同的目的地址，为此可选定一个矩阵 $\boldsymbol{H}$，$H_{ij,pq}=1$ 表示

信元(i,j)与(p,q)具有相同的目的地址，否则 $H_{ij,pq}=0$.

$$E_1=\frac{A}{2}\sum_{i=1}^{N}\sum_{j=1}^{W}\sum_{\substack{p=1\\p\neq i}}^{N}\sum_{q=1}^{W}H_{ij,pq}V_{ij}V_{pq}$$

这样，$H_{ij,pq}=1$ 时信元(i,j)与(p,q)有冲突，

$$H_{ij,pq}V_{ij}V_{pq}\neq 0$$

只有被选中的信元都不阻塞时，取最小值 $E_1=0$.

(2)每一个窗口中最多只能有一个信元被选中，神经元外部输入 I_{ij} 为 1 表示 ij 位置有信元，否则 $I_{ij}=0$.

$$E_2=\frac{B}{2}\sum_{i=1}^{N}\sum_{j=1}^{W}\left(I_{ij}-\sum_{q=1}^{W}V_{iq}\right)^2$$

当所有窗口都有信元选中时，取最小值 $E_2=0$.

(3)还有两项规定了信元发送的优先权，信元在队列中位置越靠前优先权越大，且队列越长，队列中的信元优先权也就越大.

$$E_3=-C\sum_{i=1}^{N}\sum_{j=1}^{W}(D-I_{ij})V_{ij}$$

总的目标函数为

$$E=E_1+E_2+E_3$$

4.4.4 广域网中的路由选择问题

路由选择的目标是要找到网络内两节点间的最佳路径，常用 Hopfield 神经网络来解决这类问题. 但是，由于 Hopfield 神经网络自身的特点，当它处理大型网络的路由选择问题时，常会出现无效解的情况. 所以我们引进基于分层机制的神经网络算法(Hopfield & Sparse Neural Network Algorithm，简称 HAS)来计算广域网内任意两节点间的最佳路径.

1. 对广域网进行划分并用遗传算法删除子网间的冗余连接

广域网的拓扑结构如图 4.11 所示. 因为广域网的节点数多，而且节点的分布具有区域性，所以要寻找源节点和目标节点的最佳路径，首先要按照组内节点间连接数较多、距离较近而组间节点连接较少的规则把广域网在逻辑上划分为许多子网. 在广域网中连接两个子网的节点，称为桥节点. 一个子网内两个桥节点间的最佳路径，称为最优桥路径.

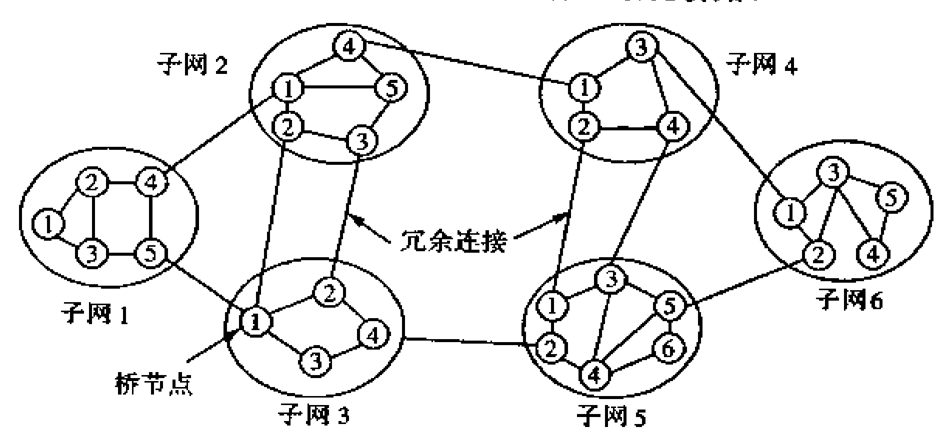


图 4.11　广域网的拓扑结构

如果广域网中的两个子网有多于一条的连接就只能保留一条最好的，可以用遗传算法删除子网间的冗余连接.具体的实现过程是：

先对染色体编码，并确定染色体的适应度函数.设染色体 C 为一 $N\times N$ 维矩阵，即染色体是由 N 个基因构成，每个基因又由 N 个子基因构成.设第 i 个基因的第 j 个子基因 C_{ij} 为 k，$k=0$ 到 $q-1$ 间的随机数（q 为子网 i 和子网 j 间的连接数），则这两子网间保留第 k 条连接，并且删除它们的冗余连接.例如，“$-1,1,2,-1,-1,-1$”是 6×6 矩阵的染色体 C 的第一个基因，它代表图 4.11 所示的广域网中第一个子网的连接情况.$C_{12}=1$ 表示子网 1 和子网 2 间保留第一条连接并且删除它们的冗余连接.$C_{14}=-1$，表示子网 1 和子网 4 无连接.由这样一个染色体 $C\{C_{ij}\}$ 在逻辑上定义了一个化简的广域网，该网中任意一对子网间至多有一条连接，并用 $N\times N$ 维矩阵 $W\{W_{ij}\}$ 表示该广域网中各子网间的连接耗散值.W_{ij} 表示子网 i 和子网 j 间保留下来的冗余连接耗散值.染色体的适应度函数为

$$f=1-\frac{1}{m}\sum_{i=1}^{m}\varphi(a_i,b_i) \tag{4.4.13}$$

其中

$$a_i=L_1(O_i,D_i),\quad b_i=L_2(O_i,D_i)$$

节点 O_i 和节点 D_i 是广域网中的任意两个节点，并且 O_i 是源节点，D_i 是目标节点.用 dijkstra 方法计算 O_i 和 D_i 在原广域网中的最佳路径函数 $L_1(\cdot)$，O_i 和 D_i 在化简后的广域网中的最佳路径函数 $L_2(\cdot)$ 及两种最佳路径的相对误差值计算函数 $\varphi(\cdot)$.最终结果与 m 次误差的平均值成反比，即若染色体适应度越高，则化简的广域网越好，也越接近最佳路径.

2. 子网内的最优路径用 Hopfield 神经网络计算，而子网间的最优路径用双向的稀疏神经网络计算

子网内最优路径的计算包括：计算子网内任意两节点间的最优路径，以及计算子网内两桥节点间的最优路径.因为子网内节点数少，所以用 Hopfield 神经网络计算子网内最优路径.为了加快整个网络最优路径的计算速度，每个子网内指定一节点定期计算本子网各桥节点间的最优路径，并将其传播到其他子网的节点.

由最小选择模型(SSM)构成的双向稀疏神经网络可用来计算无向图，即实际网络子网间的最短路径.SSM 是一些阈值型的或线型的神经元组成的模型，它能够从 n 个输入中找到最小的输入，并将其传到输出.

双向稀疏神经网络的结构如图 4.12 所示，图中每个内有数字的大圈是最小选择模型 SSM，它代表广域网中的一个子网.两个大圈间的一对双向箭头表示子网间的一条无向连接.这样，这个双向稀疏神经网络就能代表化简的广域网中各个子网间的拓扑结构（广域网内各子网间至多有一条连接）.对于每个子网，指向子网的箭头为输入，离开子网的箭头为输出.这样每个子网可以从它的所有输入中找到最小值，并将此最小值传到它的所有输出.

图 4.12 中的内有“+”的小圈，它们是一种求和神经元，如图 4.13 所示.转移函数为分段线型，即当 $X>0$ 时，$Y=X$；当 $X\leqslant 0$ 时，$Y=0$.求和神经元位于子网 j 和子网 k 间，它通过两个有向箭头与两子网连接.它有 3 个输入：X_j，W_{jk} 和 D_{ijk}.X_j 是子网 j 的输出，W_{jk} 是子网 j 与子网 k 的连接耗散值，D_{ijk} 是子网 j 内连接于网 i 和子网 k 的最优桥路径，其中子网 i 对子网 j 的输出是子网 j 的最小输入值.一般情况下，这 3 个输入的权值为 +1.如果子网 i 传出的路径不是合理路径（由源节点出发的路径为合理路径），则输入 X_j 的权值为 $+\infty$，即当路径不是合理

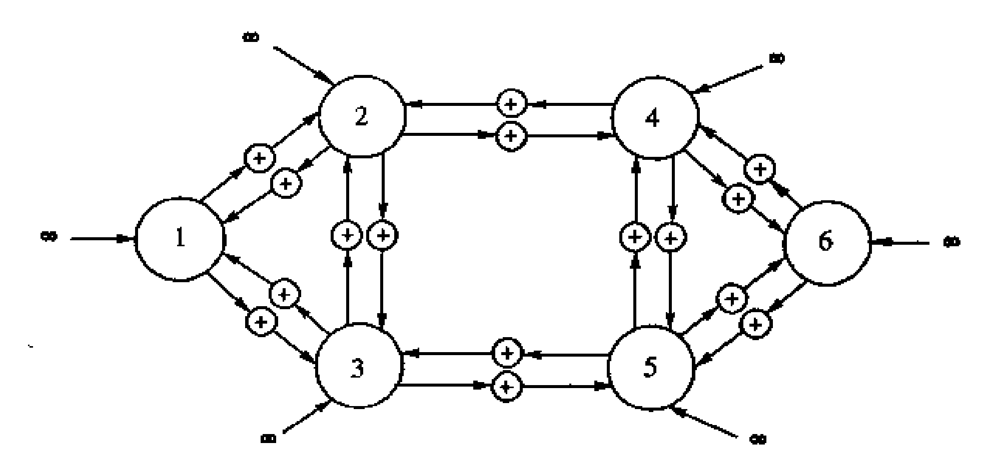


图 4.12　双向稀疏神经网络

路径时就将其屏蔽掉.

从图 4.13 中可以看出每个子网有一个特殊的输入,它没有上一级子网,且它的输入值被定为+∞,这个输入被称为独立输入. 在双向稀疏神经网络运算时,独立输入能保证合理路径的产生.

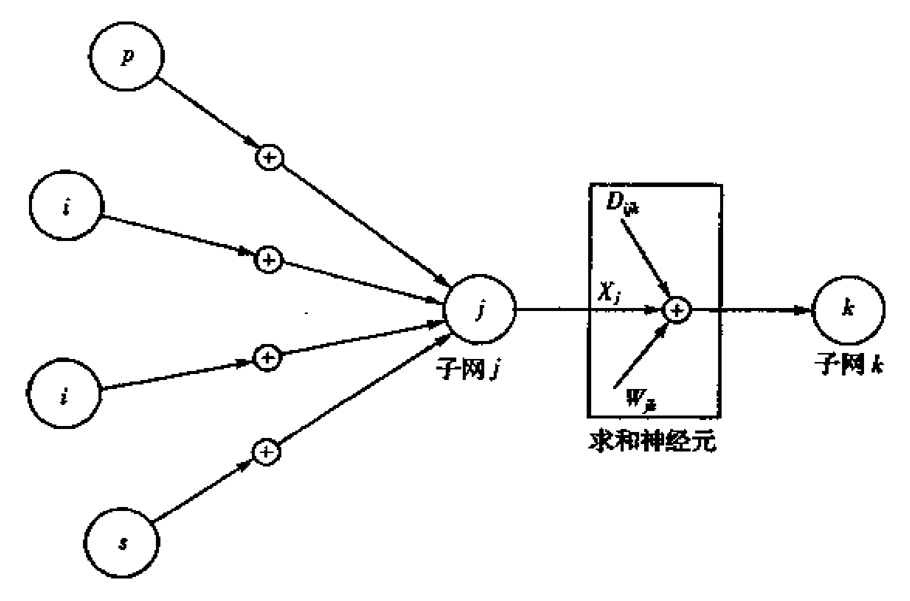


图 4.13　求和神经元的结构

3. 用 HSA 算法计算广域网中任意两节点的最优路径

假定广域网中有 n 个子网,而且化简后各子网间至多有一条连接. 用 $n\times n$ 维矩阵 W 记最 n 个子网间连接耗散值.

(1)保存矩阵 W.

(2)化简源和目标子网:设源节点 i 在源子网 k 中,子网 l 与子网 k 相连,且源子网 k 中与子网 l 相连的是桥节点 j. 用 Hopfield 神经网络计算源子网 k 中源节点 i 与桥节点 j 的最优路长 L_{ij}. 将源子网 k 与子网 l 间的连接耗散值 W_{kl} 用 $W_{kl}+L_{ij}$ 代替. 由于 W 矩阵的对称性,修改 W_{lk},使 $W_{lk}=W_{kl}$. 其他与源子网相连的子网也做如此操作. 这样,源子网就被化简成只包含一个节点(源节点). 对目标子网也做如此操作,使得目标子网只包含目标节点.

(3)根据化简的广域网构造一个双向稀疏神经网络.

(4)将源子网独立输入值及源子网内各条路径长度均设为 0. 如果子网的地名链用于记录

从源子网到该子网所走路径，在此步骤将给每个子网建立一条地名链，这 n 个地名链初始为空.

(5)按编号顺序从第 1 号到第 n 号将各子网异步串行操作.每个子网从所有输入中找到最小输入，并把它传到所有输出.如果此最小输入所代表的路径是合理路径，就将下一级求和神经元的输入 X_j 的权值设为 $+1$，将本子网的地名链复制为最小输入上级子网的地名链，并将子网代号写入此地名链.例如，子网 6 的最小输入是子网 4 的最小输出，而且子网 4 的地名链为(1,2,4)，即从源子网 1 到子网 4 的最佳路径为子网 1 经过子网 2 到达子网 4，则子网 6 复制子网 4 的地名链，并将本子网代号 6 写入自己的地名链，子网 6 的新地名链为(1,2,4,6).如果此输入所代表的路径是不合理路径，将下一级求和神经元的输入 X_j 的权值设为 $+\infty$，循环执行步骤(5)，直至目标子网输出最优解或超过计算时间.

(6)恢复矩阵 W.

从求和神经元的计算方式上可以看出，HSA 算法计算子网间的最优路径时，不仅考虑到子网间的连接耗散值，还考虑到子网内桥路径的长度.

经过这些操作，各个子网的输出都是从源子网到该子网的最佳路径，即总耗散值最小的路径.因为源子网只包含源节点，目标子网只包含目标节点，所以目标子网的地名链是广域网范围内的源节点和目标节点的最优路径.

§ 4.5　Boltzmann 机

1984 年由 Hinton G H 等人借助统计物理学的概念和方法，提出了 Boltzmann 机模型.在节点的状态变化中引入了概率和隐节点，并用模拟退火算法(Simulated Annealing，SA)进行学习.

4.5.1　Boltzmann 机的网络模型

一、网络结构

Boltzmann 机是一种有反馈的相互结合型网络，假定网络由 n 个节点构成，任意一个节点 i 的输出为：$v_i(t)\in\{0,1\}$，且假定节点之间的连接权值矩阵是对称的，即：$w_{ij}=w_{ji}$，$w_{ii}=0$.当节点 i 的输入

$$u_i(t)=\sum_{i\neq j}w_{ij}v_j(t)-\theta_i$$

发生变化时，将引起输出 $v_i(t+1)$ 的更新.这种更新在各节点之间是以异步方式进行的，并且以概率 P_i 值来决定 $v_i(t+1)$.在此规定 $v_i(t+1)$ 为 1 的概率 P_i 为：

$$P_i=f(u_i(t)/T) \tag{4.5.1}$$

$$f(x)=\frac{1}{2}(1+\tanh(x/2))=$$

$$1/(1+\exp(-x)) \tag{4.5.2}$$

将式(4.5.2)代入式(4.5.1)，得到：

$$P_i=1/(1+\exp(-u_i(t)/T)) \tag{4.5.3}$$

T 称为网络温度，T 的变化将影响函数 $f(x)$ 的形状，即决定 $f(x)$ 的曲率.如图 4.14 所示，当输入值增大时，状态为 1 的概率将增高.同时 P_i 还和 T 值有关，T 大时，曲线变化较平缓，当 T

$\to\infty$时，P_i值将为 0.5；当 T 变小时，曲线变陡，当 $T\to 0$ 时，$f(x)$变为阈值函数. 由此可见，Hopfield 网络是 Boltzmann 机的特例.

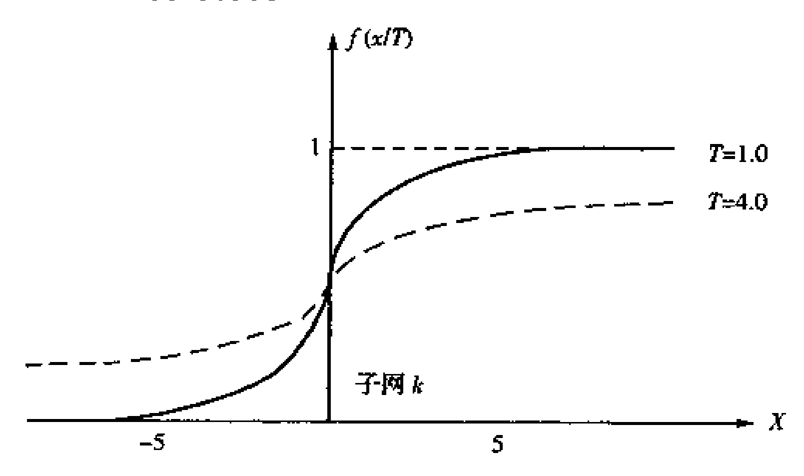


图 4.14　节点的输出受温度 T 的影响

二、网络能量

网络的能量函数定义为：

$$E(t)=-\frac{1}{2}\sum_i\sum_{j\neq i}w_{ij}v_i(t)v_j(t)-\sum_i\theta_i v_i(t) \tag{4.5.4}$$

根据式(4.5.3)给定的概率，考察 $v_i(t+1)$更新为 1 时，网络能量的变化为：

$$E(t+1)-E(t)=-(1-v_i(t))u_i(t) \tag{4.5.5}$$

由上式可知，当 $u_i(t)\geqslant 0$ 时，网络能量将减少或不变，且 $u_i(t)\geqslant 0$ 时，概率 P_i大；相反，$u_i(t)<0$ 时，网络的能量增加或不变，且 $u_i(t)<0$ 时，概率 P_i小. 总之，Boltzmann 机允许网络状态以小的概率往能量大的方向迁移，这样做的目的是使网络能量函数能够收敛于全局最小点.

三、Boltzmann 机网络状态变化规则

①从 n 个节点中随机地选取节点 i；

②计算节点 $i(1\leqslant i\leqslant n)$的输入 $u_i(t)$，

$$u_i(t)=\sum_{j\neq i}w_{ij}v_j-\theta_i$$

③以概率 P_i来决定 i 的输出 $v_i(t+1)$，

$$P_i=f(u_i(t)/T)$$
$$f(x)=1/(1+\exp(-x))$$

④其他节点不变化

$$v_j(t+1)=v_j(t)\qquad \forall j\neq i$$

⑤返回①.

4.5.2　模拟退火算法

一、模拟退火算法(SA)

模拟退火算法是基于固态物质的退火过程，通常先将它加温，使其中的粒子能够自由移动，然后逐渐降低温度，粒子也逐渐形成低能态的晶格. 若在凝结点附近温度的下降速度足够慢，则固态物质一定会形成最低能量的基态.

SA 的特点是退用性强和可达全局最小，而梯度下降算法总是向改进解的方向搜索，这种

“贪心”的算法往往导致只能找到一个局部最优解，而不是全局最优解，如图 4.15 所示. 在 SA 算法中，在系统能量减小这样一个总的趋势过程中，允许搜索偶尔向能量增加的方向搜索，以避开局部极小，而最终能够稳定到全局最优状态.

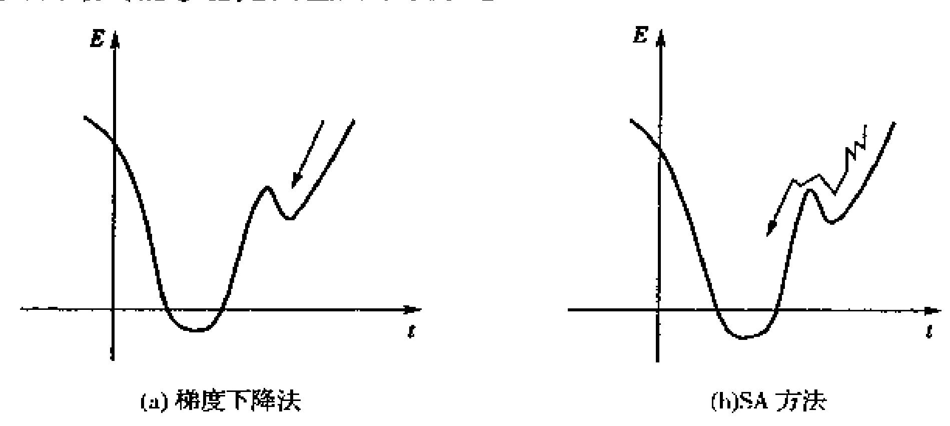


图 4.15 能量最小点的搜索

二、Boltzmann 分布

Boltzmann 机由于采用 SA 算法，并对状态迁移引入了概率的概念，即网络的状态遵从状态变化规则不断迁移，最终按不同的概率收敛于不同的状态，所以，Boltzmann 机收敛到各个状态的概率分布恰好和 Boltzmann 分布相一致. Botlzmann 分布函数是关于状态能量的函数：

$$Q(E_n) = (1/Z)\exp(-E_n/k_B T) \tag{4.5.6}$$

$$Z = \sum_n \exp(-E_n/k_B T) \tag{4.5.7}$$

其中，Z 是概率分布规格化常数，T 是状态变化规则所用的温度参数，E_n 是状态 n 的能量，k_B 为波耳兹曼常数. 从上式可以看出，网络往能量低的状态迁移的概率大，往最小能量状态迁移有最大的概率. 另外，从上式还可以看出，概率 $Q(E_n)$ 随着温度 T 的降低对能量越小越敏感，并且当 $T\to 0$ 时，最小能量状态实现的概率为 1.0，其他的状态实现的概率均为 0. 利用这一点，网络的状态能够收敛于能量函数的最小点.

在 SA 算法中，开始必须从高温出发，达到某种平衡状态后，再破坏平衡状态让温度徐徐下降，最终温度达到极限 0. 显然，温度下降太快，可能仍滞留于局部极小点. Geman S 和 Geman D 曾证明，如果按下式条件降低温度，网络的状态一定会收敛于能量最小点的状态.

$$T(t) \geqslant C/\lg(1+t) \tag{4.5.8}$$

其中，t 是状态规则的变化次数，C 是一常数. 但是按此式变化温度，由于过于缓慢，而使计算时间大大增加. 因此，在使用 SA 算法时，一些旨在提高速度的方法被提出.

图 4.16 为模拟有 4 个节点的 Boltzmann 机的实验结果. 为加快其执行速度，温度 T 的下降使用状态变化规则次数的倒数.

图 4.16(a)是初始温度为 100，终止温度也为 100，使用 10000 次状态变化规则时的输出结果. 将 16 个状态出现的频度 Prob(actual)和按式(4.5.6)用 Boltzmann 分布理论计算出的状态出现概率 Prob(theory)进行比较，可以看出二者比较接近. 如图 4.16(b)，如果将温度从 100 降至 0.01，能量最小的状态 4 的概率 Prob(actual)为 1.0.

Time	Temperature		
0.0	100.000000000		
10000.0	100.000000000		

State		Energy	Prob(theory)	Prob(actual:0)
0	:	0	0.066786738	0.069100000
1	:	244	0.005821189	0.004800000
2	:	236	0.006306019	0.007300000
3	:	588	0.000186655	0.000100000
4	:	−150	0.299317394	0.311700000
5	:	−17	0.079162645	0.076200000
6	:	315	0.002861954	0.003700000
7	:	556	0.000257047	0.000200000
8	:	−93	0.169271600	0.153400000
9	:	183	0.010713499	0.011800000
a	:	259	0.005010344	0.007500000
b	:	643	0.000107690	0.000000000
c	:	−149	0.296339136	0.297700000
d	:	16	0.056911904	0.054900000
e	:	432	0.000888256	0.001500000
f	:	705	0.000057931	0.000100000

(a)

Time	Temperature		
0.0	100.000000000		
10000.0	0.010000000		

State		Energy	Prob(theory)	Prob(actual:4)
0	:	0	0.000000000	0.000000000
1	:	244	0.000000000	0.000000000
2	:	236	0.000000000	0.000000000
3	:	588	0.000000000	0.000000000
4	:	−150	1.000000000	1.000000000
5	:	−17	0.000000000	0.000000000
6	:	315	0.000000000	0.000000000
7	:	556	0.000000000	0.000000000
8	:	−93	0.000000000	0.000000000
9	:	183	0.000000000	0.000000000
a	:	259	0.000000000	0.000000000
b	:	643	0.000000000	0.000000000
c	:	−149	0.000000000	0.000000000
d	:	16	0.000000000	0.000000000
e	:	432	0.000000000	0.000000000
f	:	705	0.000000000	0.000000000

(b)

图 4.16　SA 算法的模拟结果

4.5.3　Boltzmann 机的学习算法

使 Boltzmann 机遵从状态变化规则进行更新，平衡状态中各状态出现的概率收敛于用 Boltzmann 分布所表示的平衡状态. 而各状态的能量是由节点间的连接权值和阈值来决定，因而适当调节网络参数能够实现所希望的平衡分布，这就相当于 Boltzmann 机的学习. 学习的目的是要通过环境给出的一组范例(一组已知 I/O 值的例子)，求出 Boltzmann 机中各节点间的连接权值.

Boltzmann 机的学习可分为自我回忆型和相互回忆型两种，其形态和功能虽有差异，但学

习算法是相似的. 如图 4.17 所示，图(a)为自我回忆型，将节点分为可视节点集(Visible Units)和隐节点集(Hidden Units)两部分. 学习过程为不断调整权值 w_{ij}，使可视节点集合中的状态的平衡分布和所希望的概率分布相一致，即尽可能地逼近外部模式的概率分布. 图(b)为相互回忆型，节点分为输入节点集、输出节点集和隐节点集三部分. 当固定输入集状态时，使输出集的平衡分布和所希望的概率分布相一致. 输入集状态(相当于输入模式)和输出集状态(相当于输出模式)间存在对应关系. 例如，对输入集固定为表示"苹果"的模式时，在输出集对应为"红的"、"圆的"等多个模式，以一定的概率出现，这种相互回忆型能够实现联想记忆.

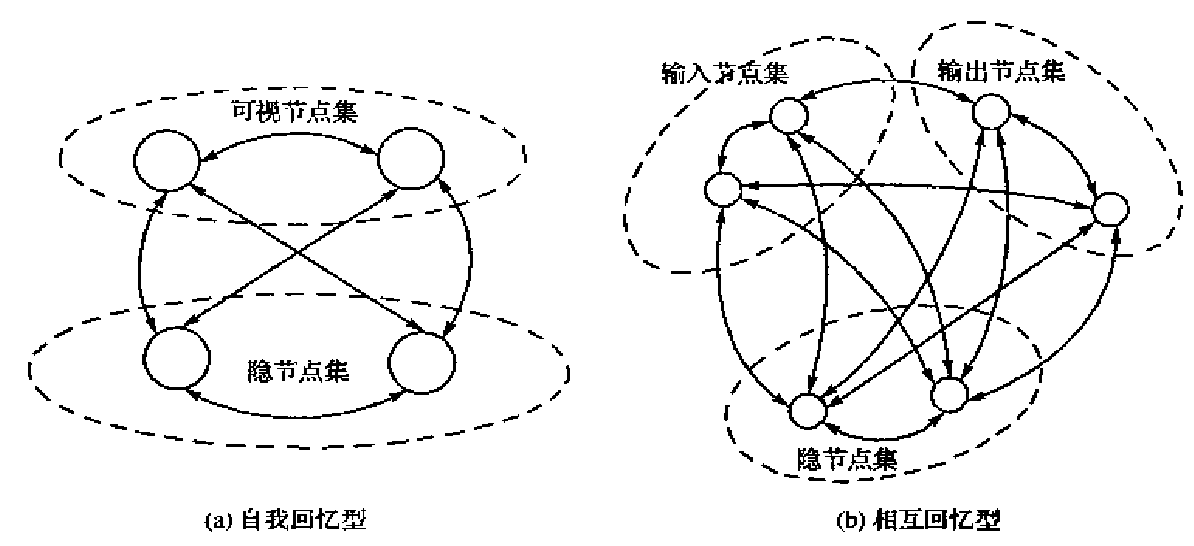


图 4.17　Boltzmann 机的学习算法

下面介绍自我回忆型算法. 假定可视节点集的状态为 V_a，节点数为 n，则 $V_a \in \{0,1\}^n$，同样，隐节点集的状态设为 H_b，节点数为 m，则 $H_b \in \{0,1\}^m$，其中 a、b 为状态号. $P(V_a)$ 为可视节点集所希望的分布函数，即外部模式的概率分布. 网络全部节点的平衡分布为 $Q_W(V_a, H_b)$，W 表示网络参数，网络状态 (V_a, H_b) 的能量用 $E_W(V_a, H_b)$ 表示，则 $Q_w(V_a, H_b)$ 能够用下面的 Boltzmann 分布给出：

$$Q_W(V_a, H_b) = (1/Z)\exp(-E_W(V_a, H_b)/k_B T) \tag{4.5.9}$$

$$Z = \sum_{a,b} \exp(-E_W(V_a, H_b)/k_B T) \tag{4.5.10}$$

可视节点集的状态分布函数 $Q_W(V_a)$ 为：

$$Q_W(V_a) = \sum_b Q_W(V_a, H_b) \tag{4.5.11}$$

此时，所希望的分布 $P(V_a)$ 和网络所实现的分布 $Q_W(V_a)$ 间的差异用 $G(W)$ 来表示：

$$G(W) = \sum_a P(V_a)\lg[P(V_a)/Q_W(V_a)] \tag{4.5.12}$$

$G(W)$ 称为 Kullback 信息量，表示分布函数间的距离. 对于任何 $Q_W(V_a)$，$G(W) \geqslant 0$，当 $P(V_a) = Q_W(V_a)$ 时，$G(W)=0$ 成立. 因而两络的学习过程相当于求 $G(W)$ 的极小值，而 $G(W)$ 又由 W 值决定，W 的微小变化量 ΔW 的变化使 $G(W)$ 的变化量为：

$$G(W + \Delta W) = G(W) + \Delta W(\partial G/\partial W) \tag{4.5.13}$$

ε 为一小常数，设：

$$\Delta W = -\varepsilon(\partial G/\partial W) \tag{4.5.14}$$

$G(W)$ 一定减小，即：

$$G(W+\Delta W)\leqslant G(W) \tag{4.5.15}$$

因而，W 从给定的初始值开始，使用式(4.5.14)不断地修正，将使 $G(W)$ 趋于极小值. 在式(4.5.14)中 G 对 W 的偏微分，由下式给出：

$$\left(\frac{\partial G}{\partial w_{ij}}\right)=-\left(\frac{1}{T}\right)(P_{ij}^{(+)}-P_{ij}^{(-)}) \tag{4.5.16}$$

$$P_{ij}^{(+)}=\sum_a P(V_a)\frac{\sum_b V_iV_j\exp(-E_W(V_a,H_b)/k_BT)}{\sum_b \exp(-E_W(V_a,H_b)/k_BT)} \tag{4.5.17}$$

$$P_{ij}^{(-)}=(1/Z)\sum_{a,b}V_iV_j\exp(-E_W(V_a,H_b)/k_BT) \tag{4.5.18}$$

$P_{ij}^{(+)}$ 是可视节点集的输出值遵照所希望的分布 $P(V_a)$ 来固定，仅是隐节点集的节点按状态变化规则实现更新，当达到平衡状态时，节点 i 和节点 j 的输出同时为 1 的期望值. $P_{ij}^{(-)}$ 是网络节点全部更新时，当达到平衡状态时，节点 i 和节点 j 同时输出为 1 的期望值. 将这些式子代入式(4.5.14)能求出节点间权值的修正量：

$$\Delta W_{ij}=(\varepsilon/T)(P_{ij}^{(+)}-P_{ij}^{(-)}) \tag{4.5.19}$$

式中右边第一项是和节点 i、节点 j 的输出相关，是按比例增大权值的项，和 Hebb 规则相类似. 右边第二项是按比例减小权值的项，称做反学习(unlearning). 总之，网络通过可视节点集和外部环境相接触时，进行 Hebb 学习，当切断和外部环境的联系时(睡眠状态)，进行反学习，因此称 Boltzmann 机的学习过程为“睡眠和记忆”.

Boltzmann 机的学习算法可归纳为：

①用概率 $P(V_a)$ 固定可视节点集的各节点输出值为 V_a.

②隐节点集的状态更新到温度 T 时的平衡状态.

③全部节点对(i 和 j)在平衡状态时，计算同时输出为 1 的次数 $n_{ij}^{(+)}$.

④使可视节点集也更新，全部节点的状态更新到温度 T 时的平衡状态.

⑤全部节点对(i 和 j)在平衡状态时，计算同时输出为 1 的次数 $n_{ij}^{(-)}$.

⑥反复执行①至⑤，计算 $n_{ij}^{(+)}$ 和 $n_{ij}^{(-)}$ 的平均值来作为 $P_{ij}^{(+)}$ 和 $P_{ij}^{(-)}$.

⑦对全部节点间的权值按下式修正：

$$\Delta W_{ij}=(\varepsilon/T)(P_{ij}^{(+)}-P_{ij}^{(-)})$$

⑧返回①.

上面学习算法的②和④是为了实现温度 T 时的平衡状态. 遵从状态变化规则，T 从高温开始，用模拟退火算法来冷却，能够实现在稳定时温度 T 的平衡状态.

以上算法能够用程序模拟实现，但是计算量太大，为此只考虑可视节点集和隐节点集各一个节点的自我回忆型 Boltzmann 机，如图 4.18 所示. 此时，可视节点和隐节点各有两个状态. 学习的目的是实现可视节点集合所希望的分布，给出 V_0 为 1 的概率.

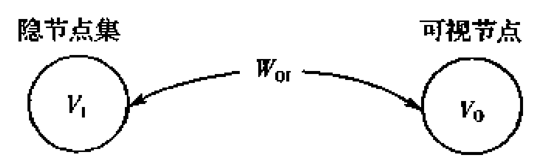


图 4.18 实验用的 Boltzmann 机

图 4.19 为 Boltzmann 机学习的例子，图(a)是使用学习规则前的初始状态，可视节点输出

为 1 时所希望的概率为 0.1. 在模拟退火算法中，温度从 100 开始，冷却到 10. 在初始状态时，网络实现的可视节点输出为 1 的概率为 0.616，和所希望的分布间的 Kullback 信息量 $G(W)$ 为 0.584769. 图(b)是在学习过程中的输出，通过不断地学习，概率变为 0.09800，$G(W)$单调减小为 0.000022.

```
《Boltzmann Machine》
 (Parameters)              :
  magnitude                :        10.000000
  ramdom—seed              :        3
  initial—state            :        0
  start—temperature        :        100.000000
  end—temperature          :        10.000000
  accumulation             :        1000
  main—loop                :        10
  sub—loop                 :        23
  epsilon                  :        0.100000
  on—probalility           :        0.100000
 [—1]
 (weights)
      8.00000          —10.00000
     —10.00000          10.00000
 (Test)
     State            Energy          Prob<theory>       Prob<act>
     [0:00]           (0.00000,       0.00000,           0.13100)
     [1:01]           (—8.00000,      0.00000,           0.25300)
     [2:10]           (—10.00000,     0.00000,           0.35800)
     [3:11]           (—8.00000,      0.00000,           0.25800)
     [on—probability]  :              0.616000
     [G—measure]       :              0.584769
                                 (a)
[0]
 [on—probability]     :      0.391000
 [G—measure]          :      0.215165
 [1]
 [on—probability]     :      0.319000
 [G—measure]          :      0.134947
 [2]
 [on—probability]     :      0.226000
 [G—measure]          :      0.054204
 [3]
 [on—probability]     :      0.226000
 [G—measure]          :      0.054204
 [4]
 [on—probability]     :      0.159000
 [G—measure]          :      0.011649
 [5]
 [on—probability]     :      0.139000
 [G—measure]          :      0.006940
 [6]
 (weights)
     16.98580          —16.76560
     —16.76560         —7.80080
```

(Test)

State	Energy	Prob<theory>	Prob<act>
[0:00]	(0.00000,	0.49219,	0.14600)
[1:01]	(−16.98580	,0.42188,	0.75600)
[2:10]	(7.80080,	0.01563,	0.04300)
[3:11]	(7.58060,	0.07031,	0.05500)
[on—probability]:		0.098000	
[G—measure]		0.000022	

(b)

图 4.19　模拟 Boltzmann 机的学习

§4.6　双向联想记忆网络

单向线性联想器实现的只是线性映射，而具有反馈功能的双向联想记忆网络(BAM)才是真正意义下的异联想记忆.

一、网络结构

双向联想记忆网络是一种两层的异联想、反馈型网络，拓朴结构如图 4.20 所示，BAM 的运行是双向的，对网络的一端输入，则可在另一端得到输出，该输出又反馈回来……如此反复，直到达到稳定. 双向联想记忆网络也分为离散型和连续型两类，这里只介绍最基本的离散型网络.

设输入层有 n 个节点，$X\in\{-1,+1\}^n$，输出层有 m 个节点，$Y\in\{-1,+1\}^m$，正向权值矩阵为 W_1，逆向权值矩阵为 W_2.

网络的状态方程为：

$$\begin{cases}X(t)=f_y(f_x(X(t-1)W_1)W_2)\\Y(t)=f_x(f_y(Y(t-1)W_2)W_1)\end{cases}\tag{4.6.1}$$

转移函数为

$$f_x(x)=f_y(x)=\mathrm{sgn}(x)\tag{4.6.2}$$

二、学习规则

采用 Hebb 学习规则，设 $X\in\{-1,+1\}^n$，$Y\in\{-1,+1\}^m$，考虑阈值为 0 的情形：

$$W_1=\alpha\sum_{K=1}^{N}X^{\mathrm{T}}Y,\qquad W_2=\alpha\sum_{K=1}^{N}Y^{\mathrm{T}}X$$

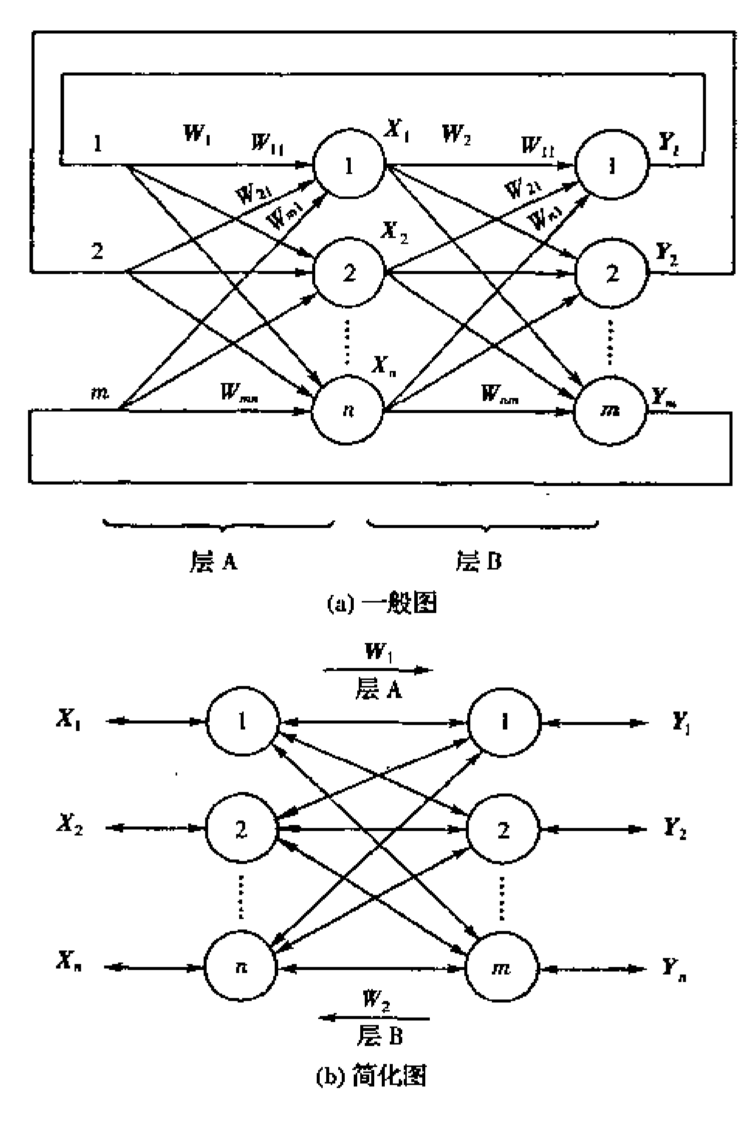
其中 α 为大于零的常数，当 $\alpha=1$，显然有 $W_1=W_2^{\mathrm{T}}$，也可以用 W 代表 W_1，用 W^{T} 代表 W_2.

三、网络的稳定性

网络的能量函数可定义为

$$E(X,Y)=-\frac{1}{2}XWY^{\mathrm{T}}-\frac{1}{2}YW^{\mathrm{T}}X^{\mathrm{T}}=-XWY^{\mathrm{T}}\tag{4.6.3}$$

可以证明由式(4.6.3)定义的能量函数随着网络状态的迁移而下降，即网络是稳定的.



(a) 一般图

(b) 简化图

图 4.20 双向联想记忆网络

§4.7 海明网络

一、网络结构

1987 年由 Lippmann 等人提出的海明(Hamming)网络是一种由上、下两层组成的网络，拓扑结构如图 4.21 所示. 设下层节点数为 n，$\boldsymbol{X}\in\{-1,+1\}^n$，上层节点数为 m，$\boldsymbol{Y}\in\{-1,+1\}^m$，两层之间权值为 w_{ij}，$i=1,2,\cdots,n$，$j=1,2,\cdots,m$，上层各单元的阈值为 θ_j. 上层各节点之间的权值为 t_{kl}，$k,l=1,2,\cdots,m$，最上层阈值取零.

转移函数定义为

$$f(x-\theta)=\begin{cases}0, & \text{当 } x<\theta\\ x-\theta, & \text{当 } \theta\leqslant x\leqslant C\\ \text{常数}, & \text{当 } x>C\end{cases} \qquad (4.7.1)$$

其中 θ 为阈值，C 是饱和值. 假定 Hamming 网络在任何情况下都不会达到饱和.

二、学习阶段

Hamming 网络的学习过程即如何确定权值 w_{ij} 和 t_{kl} 以及阈值 θ_j，以使系统达到稳定的过

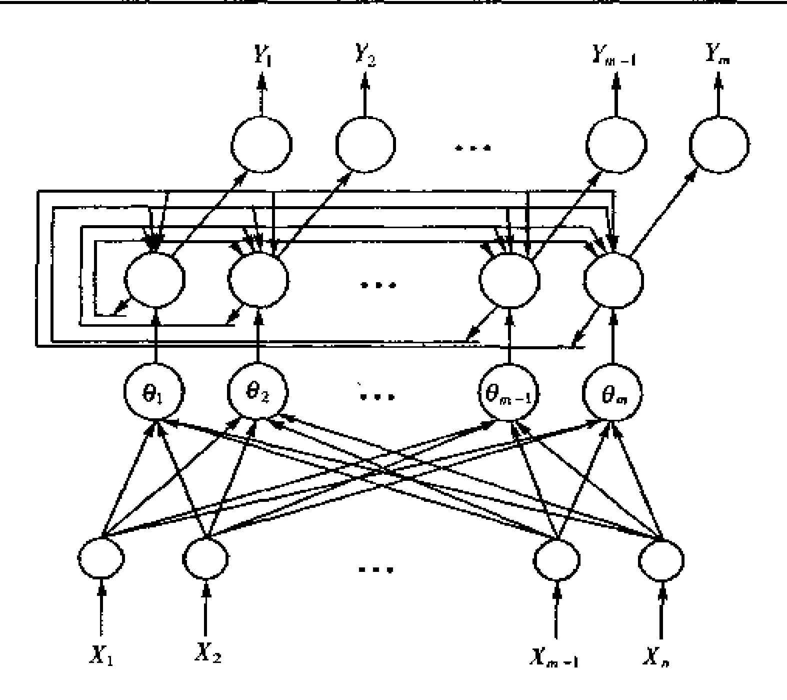


图 4.21　Hamming 网络的拓扑结构

程.

假定在网络中存入 N 个样本 $\mathbf{X}^k, k=1,2,\cdots,N$,那么上下两层单元之间的权值取为

$$w_{ij}=\frac{X_i^k}{2}\qquad(i=1,2,\cdots n;k=1,2,\cdots,N)\tag{4.7.2}$$

阈值取为

$$\theta=n/2\tag{4.7.3}$$

上层单元之间的权值取为

$$t_{kl}=\begin{cases}+1, & \text{当 } k=l\\ -\varepsilon, & \text{当 } k\neq l\end{cases}\tag{4.7.4}$$

其中 $\varepsilon<1/m, k,l=1,2,\cdots,m$,这表示上层单元之间的连接方式是自身为正反馈,相互之间为负反馈,即任一单元均有维持自身并抑制其他单元的趋势.

三、工作阶段

Hamming 网络的具体运行方式是当下层单元输入一个 $\boldsymbol{X}$ 时,通过连接 w_{ij} 输入信号到上层单元,形成上层单元的初始状态为

$$u_j(0)=f\left(\sum_{i=1}^{n}w_{ij}x_i-\theta_j\right),\qquad j=1,2,\cdots,m\tag{4.7.5}$$

上层单元按照下面的迭代方程演化,

$$u_j(t+1)=f(u_j(t)-\varepsilon\sum_{k\neq j}u_k(t))\tag{4.7.6}$$

此方程可以由式(4.7.4)给出的权值直接推导出来.上层单元的迭代过程一直进行到收敛于一个稳态为止.这是上层单元之间的一个竞争过程.注意其稳态一般只有某一个单元 j 有正的输出,而其余单元输出都为零.这可以看成是对下层单元的一种分类.

一般说来,当下层输入一个 $\boldsymbol{X}$,代表与 $\boldsymbol{X}$ 最靠近的样本的上层单元在竞争中获得胜利,同时 $\boldsymbol{X}$ 与样本要足够接近,以满足阈值条件.当下层输入为 $\boldsymbol{X}$ 时,由式(4.7.2),式(4.7.3)和式(4.7.5)可以得出上层 j 单元的输入取决于下式

$$\sum_{i=1}^{n} w_{ij} x_i - \theta_j = \frac{1}{2}\Big(\sum_{i=1}^{n} x_i^j x_i - N\Big) = \frac{1}{2}(\boldsymbol{X}^j \cdot \boldsymbol{X} - N) = - d_H(\boldsymbol{X}^j, \boldsymbol{X}) \tag{4.7.7}$$

其中 $\boldsymbol{X}^j \cdot \boldsymbol{X}$ 是 $\boldsymbol{X}$ 与 $\boldsymbol{X}^j$ 的内积，式中定义 $d_H(\boldsymbol{X}_j, \boldsymbol{X})$ 为矢量 $\boldsymbol{X}^j$ 与 $\boldsymbol{X}$ 之间的 Hamming 距离，等于两个矢量中取不同值（即相反值）的分量的数目. 两个相同矢量的 Hamming 距离为零，Hamming 距离的最大值为 n.

总之，Hamming 网络能够对离散输入模式进行在海明距离最小意义下的识别及聚类，由拓扑结构看到下层是匹配子网络，在学习阶段该子网络记忆存储样本输入模式，在工作阶段该子网络计算输入模式与样本模式的匹配程度，并把结果送入上层的竞争于网络.

第五章　自组织竞争神经网络模型

§5.1　概　　述

在实际的神经网络中,存在一种侧抑制的现象,即一个神经细胞兴奋后,通过它的分支会对周围其他神经细胞产生抑制,这种侧抑制在脊髓和海马中存在,在人眼的视网膜中也存在.这种抑制使神经细胞之间出现竞争,一个兴奋最强的细胞对周围神经细胞的抑制也强,虽然一开始各个神经细胞都处于兴奋状态,但最后是那个输出最大的神经细胞"胜"了,而其周围的神经细胞"败"了.

另外,在认知过程中除了从教师那儿得到知识外,还有一种不需要教师指导的学习,例如:婴儿出生后,听到外界声音的刺激,他自然会发出声,并自己在外界环境中学习抓东西、走路等.在动物中这种现象更为普遍.这种直接依靠外界刺激,"无师自通"达到的功能有时也称为自学习、自组织的学习方法.

自组织竞争人工神经网络就是基于上述两种生物结构和现象的基础上生成的,它的权是经过 Hebb 规则或类似 Hebb 规则学习后得到的.

§5.2　自组织特征映射网络

人脑是由巨量的神经细胞组成,它们并非起着同样的作用,处于不同空间的神经细胞分工不同,芬兰学者 Kohonen 认为一个神经网络接受外界输入模式时,将会分成不同的区域,各区域对输入模式将会有不同的响应特性,同时这一过程是自动完成的.

各神经元间的连接权值具有一定的分布,最临近的神经元互相激励,而较远的神经元之间相互抑制,而更远的神经元之间又有较弱的激励作用.如图 5.1 所示,在受到外界刺激时,刺激最强的地方形成一个 Bubble(墨西哥帽),在此 Bubble 区中,神经元权向量会自动调节,直到与输入向量的某一最大分量方向相重合为止.

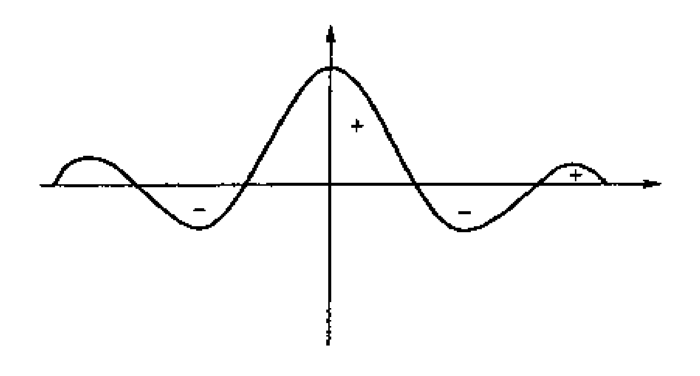


图 5.1　神经元的作用分布曲线

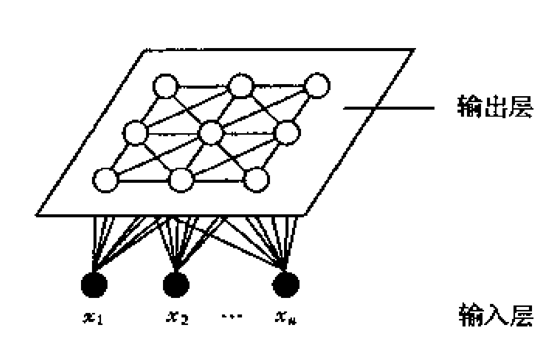


图 5.2　网络结构

5.2.1　网络拓扑结构及工作过程

自组织映射网络结构见图 5.2 所示，图中上层为输出层，假定有 m 个节点，输出层中的节点是以二维形式排成一个节点矩阵，输出节点之间也可能实现局部连接，它们中的每个节点是一个输入样本的代表. 输入层处于下方，若输入向量为 n 维，那么输入节点有 n 个，输入节点与输出层的所有节点通过权值实现全互联. 在输出层竞争是这样进行的，对于获胜的节点 j，在其周围 N_j 区域内的节点在不同的程度上得到兴奋，而在 N_j 区域以外的节点都被抑制，这个区域 N_j 可以是正方形也可以是六角形，如图 5.3 所示. 区域 N_j 是时间 t 的函数，随着 t 的增加，N_j 的面积成比例缩小，最后剩下一个节点，或一组节点，它们反映一类样本的属性. 自组织映射网络为一个无教师指导的、自适应、自组织的网络.

对于任意一个输入节点 i 和输出节点 j 有：

$$i = \{1,i,\cdots,n\},j \in \{1,2,\cdots,m\} \text{ 且：} Y_j = \sum_{i=1}^{n} W_{ij}X_i \tag{5.2.1}$$

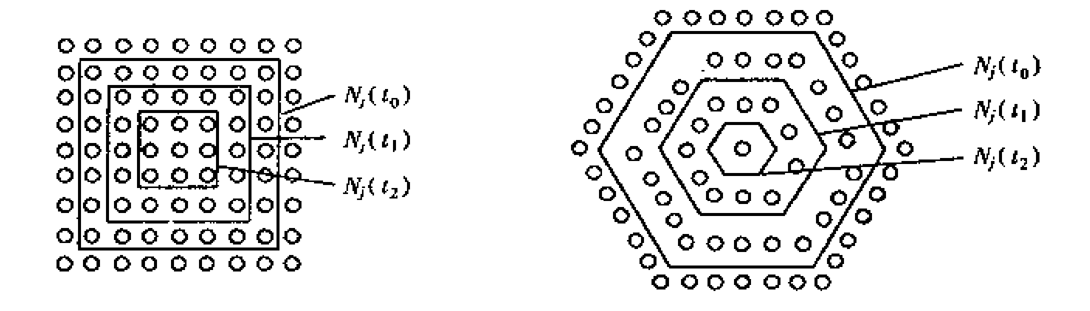


图 5.3　$N_i(t)$的形状变化情况

5.2.2　自组织映射学习算法

假定共有 K 个输入模式，对于某个输入模式通过竞争，逐渐收敛到样本空间所划分的 K 个子集的中心. 当某一模式输入时，对某一输出节点给予最大的刺激，以指示该类模式的所属区域，而同时对获胜周围的一些节点给予较大的刺激. 当另一输入模式输入时，获胜区域移到其他区域.

在训练过程中定义获胜节点的邻域为 $N_j(t)$，表示在时刻 t 以节点 N_j 为中心的某一半径内的所有节点，随着训练过程的进行 $N_j(t)$的半径将逐渐缩小，最后只剩 N_j 本身. 即在初始阶段不但对获胜节点 N_j 调整权值，也对其周围节点调整权值，直到最后仅对 N_j 调整权值. 保证对于某一类输入模式，获胜节点做出最大的响应，面相邻节点做出较大的响应. 几何上相邻的节点代表特征相近的模式.

网络的学习算法为：

①连接权值的初始化.

$$t = 0:0 < W_{ij} < 1, i \in \{1,2,\cdots,n\}, j \in \{1,2,\cdots,m\}$$

②对网络输入一个样本模式.

$$X^K = (X_1, X_2, \cdots, X_n)$$

③计算 X^K 与全部输出节点间的权值向量 $\boldsymbol{W}$ 的距离

$$d_j(t) = \sum_{i=1}^{n}(X_i - W_{ij}(t))^2, \qquad j \in \{1,2,\cdots,m\} \tag{5.2.2}$$

④选择有最小距离的节点 N_j^* 为竞争获胜节点，

$$d_j^* = \min(d_j), \qquad j \in \{1,2,\cdots,m\}$$

⑤调整权值

$$\begin{aligned} \Delta W_{ij} &= \alpha(t)\beta(N_j, N_j^*)(X_i - W_{ij}(T)), & j \in N_j^*(t) \\ \Delta W_{ij} &= 0, & j \notin N_j^*(T) \end{aligned} \tag{5.2.3}$$

其中，$0<\alpha(t)<1$，为增益函数，随着时间 t 而递减；$\beta(N_j, N_j^*)$ 为限界函数，随着 N_j^* 距离递减.

⑥若还有输入样本则转②，当所有的样本输入完，且满足：

$$\max(|W_{ij}(t+1) - W_{ij}(t)|) < \varepsilon, \qquad i \in \{1,2,\cdots,n\}, j \in \{1,2,\cdots,m\} \tag{5.2.4}$$

或者完成指定的学习次数后算法结束，否则转②.

$\alpha(t)$ 和 $N_j^*(t)$ 没有一般化的数学表达方法，凭经验选取. 初始时，$N_j^*(t)$ 选取较大，后逐渐变小，$\alpha(t)$ 开始时较大，后逐渐变为 0.

从自组织映射网络算法来看，该网络有以下特点：

(1)网络中的权值是输入样本的记忆. 如果输出节点 j 与输入层 n 个节点的连接权值向量用 W_j 表示，对应某一类样本 $\boldsymbol{X}^K$ 输入，使 $\boldsymbol{Y}_j$ 达到的匹配最大，那么 $\boldsymbol{W}_j$ 通过学习以后十分靠近 $\boldsymbol{X}^K$，因此以后当 $\boldsymbol{X}^K$ 再次输入时，$\boldsymbol{Y}_j$ 节点必定兴奋，$\boldsymbol{Y}_j$ 是样本 $\boldsymbol{X}^K$ 的代表.

(2)网络学习时对权值的调整，不只是对兴奋的节点所对应的权值进行调整，而对其周围区域 N_j 内的节点同时进行调整，因此对于在 N_j 内的节点可以代表不只是一个样本 $\boldsymbol{X}^K$，而是与 $\boldsymbol{X}^K$ 比较相近的样本都可以在 N_j 内得到反映，因此这种网络对于样本的畸变和噪声的容限大.

(3)网络学习结果使比较相近的输入样本在输出二维平面上位置也比较接近.

5.2.3　自组织映射网络的工作原理

Linsker 提出了一种具有自组织(self-organization)作用的神经网络. 这种网络在没有告知分析什么特征的情况下，甚至在目标识别不知是否正确的情况下，要让神经网络构成的知觉系统识别环境中的特殊性质.

在知觉研究中，有两个问题需要解决：一是神经机制作用于知觉输入是由什么起作用的？二是神经机制的本身是怎样产生的？

所谓的组织原理是指：组织原理能够解释知觉系统是如何发生和起作用的基本原理；由组织原理可以推断出更多详细的经验信息；由组织原理可以引导出有益的经验方式以及可以导出可能的预测.

在人类的知觉系统和视觉系统中，视觉系统受到了科学家的广泛研究，一般认为人的视觉系统有如下特征：

(1)视觉系统是哺乳动物最好的知觉系统.

(2)视觉信息是分层处理的.

(3)简单特性(如方向等)在视觉早期处理. 复杂特性在视觉的较后时间段里处理.

(4)颜色、运动按平行方式处理.

(5)视网膜和脑皮层是分层的，层内细胞交互，层间细胞也发生交互作用.

(6)在一个生理层内，至少在视觉早期，存在对输入信号执行类似功能的细胞群体．这种细胞群体可以看做一个滤波器矩阵，每一个群体的细胞只接受有限"接受域"的输入信号并进行处理．在一个生理层内，存在多类这样的细胞群体．

(7)细胞的反映功能是非线性的．但"线性组合"是对这种非线性功能的很好的模拟．

由上述假设，尤其是具有特殊功能细胞群体的假设，Linsker 提出了一种具有自组织功能的人工神经网络．在这种自组织网中采用 Hebb 学习规则，即：若细胞 1 是细胞 2 的若干输入细胞中的一个，且当细胞 2 能动性高时，细胞 1 的能动性也趋高，则这时由细胞 1 放电引起细胞 2 放电的可能性增大．在这种自组织网中，是根据输入与输出间相关能动性的程度来修正并调整网络神经元间的联结强度．Linsker 提出的自组织人工神经网络如图 5.4 所示．

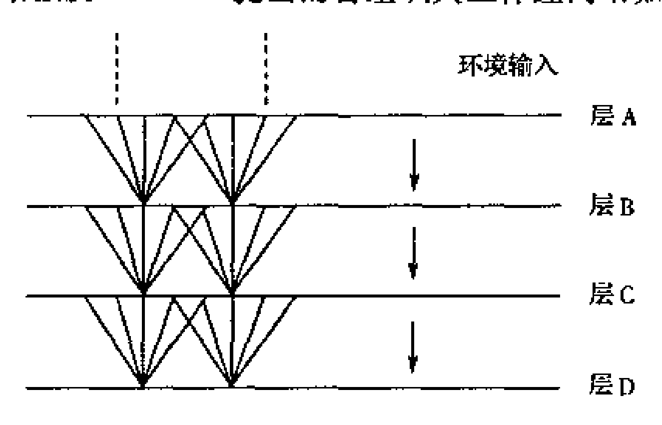


图 5.4 Linsker 自组织人工神经网络

5.2.4 网络的应用实例

Kohonen 自组织特征映射网络的应用十分广泛，例如在机器人视觉、字母的拓扑排序、语音识别、向量量化以及组合优化等领域得到了应用．

一、机器人的视觉问题

Kohonen 网络的输出节点构成三维立体数组，把机械手相对于空间中某一点的位置作为网络的输入参量．这些输入参量就决定了一个机械手在某一空间的状态．于是，可以用这些状态训练网络，使用网络的三维立体输出的状况反映该机械手的有效动作及应回避的障碍物．

二、字母的拓扑排序

假设输入有 26 个英文字母和 1～6 的六个数字共 32 个样本，每个符号对应一个五维矢量 $\boldsymbol{X}$，各个符号与相应矢量 $\boldsymbol{X}$ 的五个分量的对应关系如表 5.1．如果按照这些符号所对应的输入矢量的相似度（表示它们在一个 5 维空间中的接近程度），把这些符号画在一个树形图中，由表 5.1 看到 A，B，C，D，E 为一类，因为 $x_1=x_2=x_3=x_4=0$，F，G，H，I，J 是另一类，x_0 都为 3 外，$x_3=x_4=0$，这样就可根据表 5.1 得到图 5.5 所示的树形结构．

表 5.1

符号	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	1	2	3	4	5	6
x_0	1	2	3	4	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
x_1	0	0	0	0	0	1	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
x_2	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	3	3	3	3	6	6	6	6	6	6	6	6	6	6	6	6
x_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	1	2	3	4	2	2	2	2	2	2
x_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6

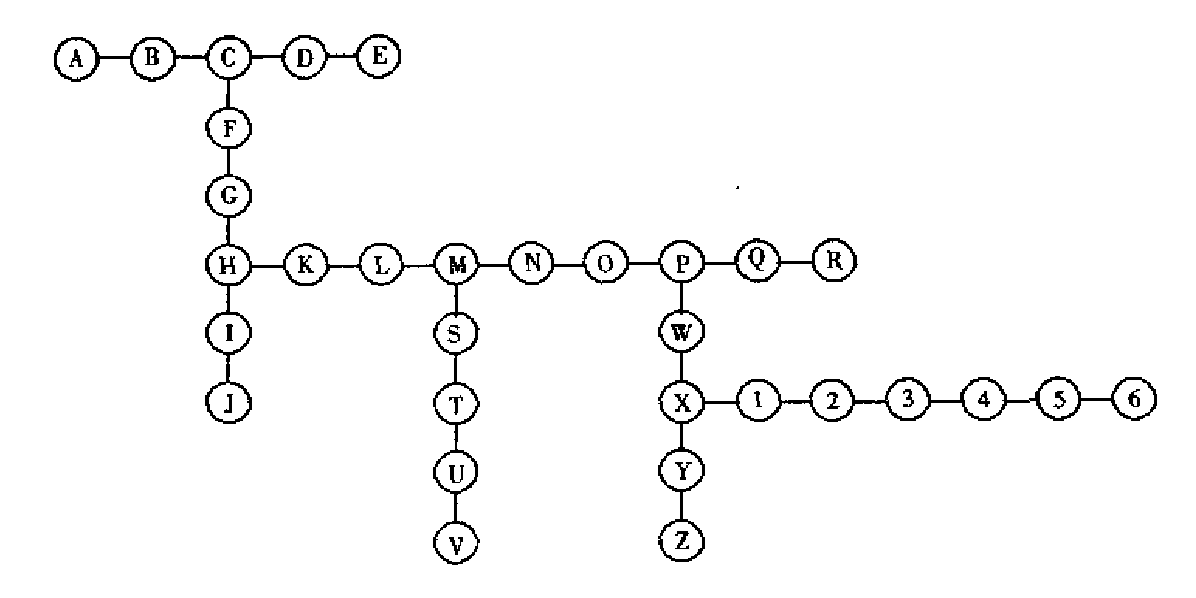


图 5.5 反映表 5.1 中 32 个符号相似关系的树形结构

现在将与各个符号对应的 $\boldsymbol{X}$ 随机地选取后送入一个二维六角形神经元阵列进行学习.完成学习后,对于每一个符号神经元阵列中有一个对其最敏感的神经元,就是说,当输入某个符号的矢量 $\boldsymbol{X}$ 时,与其相应神经元的输入加权和较其他神经元都大,这种对应关系如图 5.6 所示.

图 5.6 是通过自组织学习后得到的结果,其中 * 表示对任何输入样本都不会发生兴奋,输入的字母没有语义,但经自组织学习后形成比较有规则的拓扑结构,A、B、C、D、E 为一类,它们的 $x_1=x_2=x_3=x_4=0$,F、G、H、I、J 为一类,由图 5.6 可见,在神经元阵列中各符号之间的空间关系与它们在树状结构中的相互空间关系相当类似,两者之间拓扑结构的一致性是明显的,请注意图 5.6 上特征的矩形排列(阵列)是由 70 个神经元组成的,每一个用 5 个权和模式分量 x_0,x_1,x_2,x_3 和 x_4 相联,用从训练集中随机地选取向量 $\boldsymbol{X}_A\cdots X_6$ 对阵列进行训练,经过 10000 步训练,这些权稳定下来,而后,所得到的这个网络是已校准的.校准通过阵列神经元随来自于训练集的特定的已知向量而有指导地贴标号来实现,那么,当向量 $\boldsymbol{B}$ 是输入时,由图 5.6 可见,左上角神经元在整个阵列中产生最强的响应,所以它被标为 $\boldsymbol{B}$ 等,由图可见,32 个神经元装备有标号和 38 个神经元保留自由的.

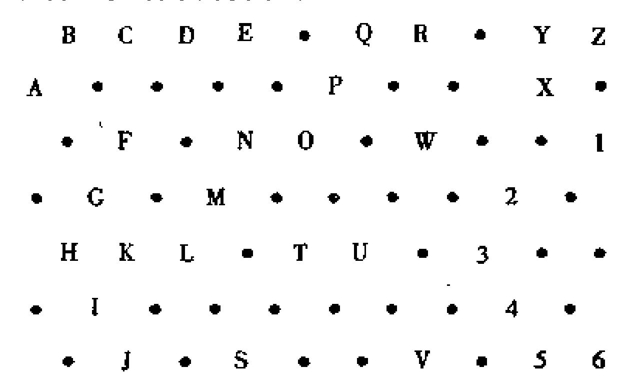


图 5.6 自组织特征映射网络输出对应于输入样本的映射

训练向量的相似度的检验揭示了 $X_A\cdots\boldsymbol{X}_6$ 能够被排列在所谓的最小间距树中,这种最小间距是 1972 年 Andrews 作为研究数据的聚类或邻域的一种技术,这种方法起因于任意数据集的一种图理论分析.最小间距树被定义为连接数据集中所有点到它们最相邻的平凡树,结果

这整个树的长度得到最小，图的每个顶点相应于来自数据集中的惟一的一个元素. 如果我们赋一个数到每个图的边即等于边连接的顶点之间的距离，那么一个集合的最小间距树是有最小值的树的边之和的间距树.

三、旅行商(TSP)问题

在 Kohonen 网络中，利用自组织学习方法在输出神经元空间中的拓扑映射. 把 N 个城市看成是二维平面上的 N 个点，那么，这些城市所处的位置作为网络的输入，把输入样本看做是一个城市的集合，其中每一个城市是定位在二维平面上的，如同地图上表示城市的位置坐标一样，在集合 C 中，每个样本用 $C_j(x_j, y_j)$来表示，城市坐标 x, y 用两个神经元作为这个网络的输入来表示，如果这种网络的输出有 100 个以上的神经元，可以组成一个平面，网络从输入到输出的连接权反映了该城市在路径中的位置，网络中的连接权开始时是一个随机数，从一个城市到另一个城市的优选原则是在某种局部限制条件下，这两个城市的距离较为接近，这样反复选择，最后得到的旅行路径不是最优也是较优的路径.

在 Kohonen 网络中，按照前面介绍的算法进行学习，其邻域 N_c 一开始选得很大，随时间减小，也随之减小. 由于 Kohonen 网络中的聚类和映射特性，随着自组织过程的进行，使得在地图上相近的城市，在网络输出平面中的位置也比较相近，把相近的点连成链状，这个链无疑是按照城市远近排列，这个链便是 TSP 的解了，具体如图 5.7.

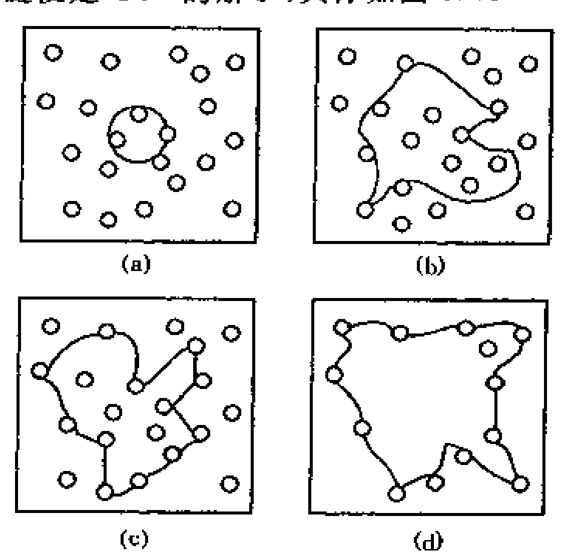


图 5.7 Kohonen 网络求解 TSP 问题的示意图

实质上，这个例子可看成为二维平面上的点到一曲线上的特征映射. 图 5.7 的运行过程就好像拉动一条橡皮圈，在自组织过程中，这个橡皮圈是决定拉向另一个输出节点，还是保持不变. 图 5.7(a)表示初始橡皮圈的位置，图 5.7(b)和(c)表示拉动的某些中间过程，而 5.7(d)给出最后结果.

在拉动过程中，拉动方向(权值更新)是由下式决定：

$$\Delta\omega_i = \eta\Big[\sum_{\mu}\beta_\mu(i)(x^\mu - \omega_i) + k(\omega_{i+1} - 2\omega_i + \omega_{i-1}\Big]$$

其中 $\eta>0$，为学习调整速度；$\beta_\mu(i)$为把 ω_j(决定城市的位置)拉动到城市 μ 的强度，其大小取决于某种城市概率分布；$k>0$ 是使橡皮圈在城市之间的互相拉动能保持一个完整的环状.

§5.3 自适应共振理论模型

5.3.1 自适应共振理论(ART)

自适应共振理论(Adaptive Resonance Theory,简称 ART)是由美国 Boston 大学 Grossberg S 和 Carpenter A 提出的.这一理论包括 ART1 和 ART2 两种模型,可以对任意多和任意复杂的二维模式进行自组织、自稳定和大规模并行处理.前者用于二进制输入,后者用于连续信号输入.

如图 5.8 所示,它由两个相继连接的存储单元 STM-F_1 和 STM-F_2 组成,分成注意子系统和取向子系统.F_1 和 F_2 之间的连接通路为自适应长期记忆(LTM).

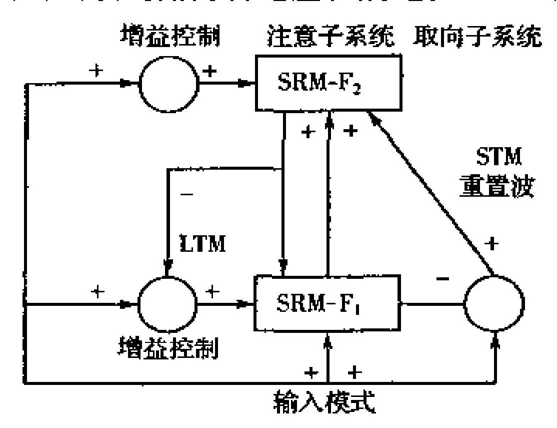


图 5.8　ART 原理图

ART 的主要优点:

①可完成实时学习,且可适应非平稳的环境;

②对已学习过的对象具有稳定的快速识别能力,同时又能迅速适应学习的新对象;

③具有自归一能力,根据某些特征在全体中所占的比例,有时作为关键特征,有时又被当做噪声处理;

④不需要事先已知样本结果,可非监督学习;

⑤容量不受输入通道数的限制,存储对象也不要求是正交的;

⑥此系统可以完全避免陷入局部极小点的问题.

5.3.2 ART1 神经网络

一、ART1 神经网络基本结构

网络的基本结构如图 5.9 所示,由两层神经网节点构成两个子系统,分别称比较层(Compare,简称 C 层)和识别层(Recognition,简称 R 层).另外还有三种控制信号,即复位信号(简称 Reset)及两种逻辑控制信号 G_1 和 G_2.

1. C 层结构

具有 n 个节点,每个节点接受来自三个方面的信号.

信号 1:输入 $\boldsymbol{X}$ 的第 i 个分量 X_i.

信号 2:R 层第 j 个单元的自上而下返回信号 R_i.

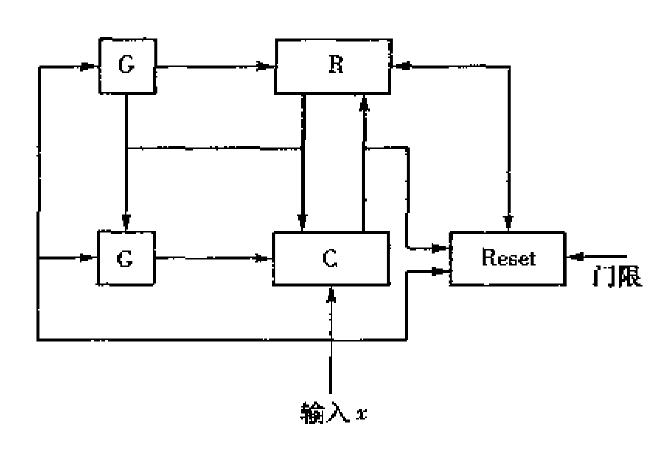


图 5.9　ART1 网络的基本结构

信号 3：G_1控制信号.

设 C 层第 i 个单元输出为 C_i. C_i 根据"2/3 规则"产生，即 C_i 具有三个信号中的多数相同的值.

网络开始运行时，$G_1=1$，R 层反馈信号为 0.

2. R 层结构

R 层功能结构相当于一种前向竞争网络，假设输出层有 m 个节点，m 类输入模式. 输出层节点能动态增长，以满足设立新模式类的需要. 设由 C 层自下而上连接到 R 层第 j 个的节点的权向量用 $\boldsymbol{W}_j=\{w_{1j},w_{2j},\cdots,w_{nj}\}$表示. C 层输出向量 $\boldsymbol{C}$ 沿 $\boldsymbol{W}_j$ 向前馈送，经过竞争在 R 层输出端产生获胜节点，指示本次输入向量类别.

3. 控制信号

(1)G_1：设输入模式 $\boldsymbol{X}$ 各元素的逻辑"或"为 x_0，R 各元素的逻辑"或非"为 R_0，则 $G_1=x_0R_0$，即只有在 R 层输出向量 $\boldsymbol{R}$ 为全 0，而输入 $\boldsymbol{X}$ 不为全 0 时，$G_1=1$，其他情况下 $G_1=0$.

(2)G_2 是输入模式 $\boldsymbol{X}$ 各元素的逻辑"或"，即 $\boldsymbol{X}$ 为全 0 时，G_2 是 0. 其他情况下 G_2 是 1.

(3)Reset：设预先设定的相似性度量为 ρ. 如按某种事先设定的测量标准，$\boldsymbol{C}$ 与 $\boldsymbol{X}$ 并非充分接近且达到 ρ，则发出 Reset 信号，以使 R 层竞争获胜节点无效. 这表示此次选择的模式代表类不能满足要求.

二、ART1 网络运行原理

从网络输入模式到最后把该模式存储至相应的模式类别中，网络的运行过程经历了三个阶段.

1. 识别阶段

在网络没有输入模式之前，网络处于等待状态. 此时，输入端 $\boldsymbol{X}=0$，并置控制信号 $G_2=G_1=0$. 所以，R 层单元输出全为 0，在竞争中有同等的获胜机会. 当网络输入不全为 0 时，置 $G_1=1$. 信息自下而上流动. 由"2/3 规则"可知，此时 C 层输出 $\boldsymbol{C}=\boldsymbol{X}$，且 $\boldsymbol{C}$ 向上馈送，与向上权向量 $\boldsymbol{W}$ 进行作用，产生向量 $\boldsymbol{T}$. $\boldsymbol{T}$ 向上送入 R 层，使 R 层内部开始竞争. 假设获胜节点为 j^*，则 R 层输出为 $R_j^*=1$，而其他节点输出为 0.

2. 比较阶段

R 层输出信息自上而下返回 C 层. R_j^* 使 R 层 j^* 节点所连接的自上而下的 $\boldsymbol{W'}_j^*$ 被激活，并向下返回 C 层.

此时，R 层输出不为全 0，并且 $G_1=0$. 所以，C 层下一次输出 $\boldsymbol{C'}$取决于由 R 层自上而下的

权向量 W'_{j^*} 及网络的输入模式 X. 即:$C'_i=W'_{j^*i}\wedge x_i$. 其中 $\wedge$ 表示逻辑与. 新状态 C' 就反映了输入向量 X 与其所激活的节点 j^* 的典型向量之间的相似度.

用事先指定的门限对相似度进行测试,若 C' 给出了足够相似的信息,则表示竞争结果正确. 反之,则表示竞争结果不符合要求,就发 Reset 信号以置上次获胜的节点无效,并使其在本次模式的匹配过程中不能再获胜. 然后进入搜索阶段.

在比较阶段,网络的信息流向是自上而下的.

3. 搜索阶段

由 Reset 信号置获胜阶段无效开始,网络进入搜索阶段. 此时 R 为全 0,$G_1=1$,在 C 层输出端又得到了本次的输入模式 X. 因此,网络又进入识别及比较阶段,得到新的获胜节点(以前的获胜节点不参加竞争). 这样重复直至搜索到某一个获胜节点 k,它与输入向量 X 充分匹配达到满足的要求为止. 模式 X 编制到 R 层 k 节点所连的模式类别中,即按一定的方法修改 k 节点的自下而上和自上而下的权向量,使网络以后再遇到 X 或与 X 相近的模式时,R 层节点 k 能很快取得竞争的胜利. 若搜索了所有的 R 层输出节点而没有发现有与 X 充分接近的模式,则增设一个 R 层节点以表示 X 或与 X 相近的模式.

三个阶段表示了对于一次外界输入,ART 网络的运行过程. 当外界输入 X 与所激活的 R 层节点的模式充分匹配时,网络发生了"共振",运行过程结束. 否则,就进行特别处理,直到发生"共振"现象时,对本次输入的训练过程才结束. 然后释放所有被 Reset 信号占用的节点,又对新的输入模式开始以上的训练过程.

三、ART1 网络学习算法

ART1 训练算法(ART1):

①初始化. 自下而上的权向量 W 赋予较小的初值,自上而下的权向量 W' 赋予初值 1. 相似度门限 $0<\rho<1$.

②始网络输入模式 $X=(x_1,x_2,\cdots,x_n)$,$x_i\in\{0,1\}$.

③若 X 不为全 0,由"2/3 规则"可知 C 层输出 $C=X$. 信息前送,由自下而上的权向量 W 进行加权,得输出为:$u_j=CW_j$,$j\in\{1,2,\cdots,m\}$.

④R 层竞争开始运行,若有 $u_j^*=\max\{u_j,j\in\{1,2,\cdots,m\}\}$,则 R 层的 j^* 节点获得竞争胜利.

⑤信息反送,由 R 层的获胜节点 j^* 送回自上而下的权向量 W'_{j^*}. 此时 $G_1=0$,由"2/3 规则"可得到 C 层新输出向量 C 的各个元索满足:$C_i=W'^{*}_{ji}x_i$.

⑥警戒线检测. 设向量 X 中不为 0 的个数用 $\|X\|$ 表示,可有

$$\|X\| = \sum_{i=1}^{n} x_i \tag{5.3.1}$$

$$\|C\| = \sum_{i=1}^{n} w_{ji}x_i \tag{5.3.2}$$

$\|C\|/\|X\|>\rho$ 若成立,则接受 j^* 为获胜节点,转⑦. 否则发 Reset 信号,置 j^* 为 0(不允许其再参加竞争),开始搜索阶段,转③.

⑦修改 R 层节点自下而上及自上而下的权向量,使其以后对与 X 相似的输入更容易获胜,且具有更高的相似性.

$$w_{ji}^*(t+1) = w_{ji}^*(t)x_i \tag{5.3.3}$$

$$w_{ij}^*(t+1)=\frac{l\times w_{ji}^*(t+1)}{l-1+\sum_i w_{ji}^*(t)x_i} \tag{5.3.4}$$

其中 l 为大于 1 的常数.

⑧恢复由 Reset 信号抑制的 R 层节点，转到②以迎接下一次的输入.

5.3.3 ART1 网络学习算法的改进

在 ART1 的训练算法中，步骤⑤得到的 C 层新输出向量 $\boldsymbol{C}$ 既不完全与 $\boldsymbol{X}$ 相同，又不完全相同于自上而下的权向量 $\boldsymbol{W'}_j^*$，所记录的记忆模式向量 $\boldsymbol{Z}=\boldsymbol{W'}_j^*$，我们记该向量为 $\boldsymbol{S}^*=\boldsymbol{C}$，$\boldsymbol{S}^*$ 所表现出的是与二者的一致之处，我们称之为综合模式向量. 如果 $\boldsymbol{Z}$ 与 $\boldsymbol{X}$ 的相似程度很高，那么 $\boldsymbol{S}^*$ 中非零分量的个数不会比 $\boldsymbol{X}$ 中非零分量的个数少许多，所以步骤⑥中的警戒线检测 $\|\boldsymbol{C}\|/\|\boldsymbol{X}\|>\rho$ 成立；反之，$\boldsymbol{S}^*$ 中非零分量的个数将明显比 $\boldsymbol{X}$ 中非零分量的个数少许多，则匹配失败.

但是，在步骤⑥警戒线检测中，显然只对综合模式与输入模式 $\boldsymbol{S}^*$ 作了比较，而没有对记忆模式 $\boldsymbol{Z}$ 与综合模式 $\boldsymbol{S}^*$ 进行比较. 如果记忆模式 $\boldsymbol{Z}$ 包含输入模式 $\boldsymbol{X}$，而它们又显然不是同一模式. 例如：$\boldsymbol{X}=\{0,1,1,0,0,0,1,0,1,0\}$，而 $\boldsymbol{Z}=\{1,1,1,1,1,1,1,1,1,1\}$，则 $\boldsymbol{S}^*=\{0,1,1,0,0,0,1,0,1,0\}$，这时，步骤⑥中的警戒线检测 $\|\boldsymbol{C}\|/\|\boldsymbol{X}\|=1$，输入模式 $\boldsymbol{X}$ 被误判为记忆模式 $\boldsymbol{Z}$.

为了解决这种误判，令：

$$\|\boldsymbol{Z}\|=\sum_{i=1}^{n}w_{ji}^* \tag{5.3.5}$$

其中 j^* 为获胜节点，则 $\|\boldsymbol{Z}\|$ 表示 j^* 节点所记忆的模式特征，即非零分量的个数；而 $\|\boldsymbol{S}^*\|=\|\boldsymbol{C}\|$，表示 $\boldsymbol{X}$ 与 $\boldsymbol{Z}$ 匹配的综合特征；$\|\boldsymbol{X}\|$ 表示输入模式特征.

引入钟形函数

$$f(x)=\frac{1}{1+(x-1)^2} \tag{5.3.6}$$

该函数当 $x=1$ 时取最大值 1.

改进的 ART1 训练算法：

ART1 训练算法其他步骤不变，步骤⑥警戒线检测作如下修改：

⑥警戒线检测. 设向量 $\boldsymbol{X}$ 中不为 0 的个数用 $\|\boldsymbol{X}\|$ 表示，可有：

$$\|\boldsymbol{X}\|=\sum_{i=1}^{n}x_i \tag{5.3.7}$$

$$\|\boldsymbol{C}\|=\sum_{i=1}^{n}w_{ji}x_i \tag{5.3.8}$$

$$\|\boldsymbol{Z}\|=\sum_{i=1}^{n}w_{ji}^* \tag{5.3.9}$$

若 $f(\|\boldsymbol{C}\|/\|\boldsymbol{Z}\|)*(\|\boldsymbol{C}\|/\|\boldsymbol{X}\|)>\rho$ 成立（函数 f 如公式(5.3.6)定义），则接受 j^* 为获胜节点，转⑦. 否则发 Reset 信号，置 j^* 为 0（不允许其再参加竞争），开始搜索阶段，转③.

5.3.4 ART2 神经网络

ART2 神经网络是为任意模拟输入矢量而设计的（二值矢量作为一种特例当然也包括在

内)，它具有十分宽广的应用范围，但不能再采用简单的“2/3 规则”，其注意子系统和调整子系统的结构要复杂得多；通过警戒参数的调整，ART2 可以按任何精度对输入的模拟观察矢量进行分类，可以用于语音识别及生成、图像处理及识别、机器人、雷达信号分类及判决等许多领域. 在给出 ART2 的设计模型并阐述其工作原理之前，我们首先讨论对于系统的总体要求，着重说明由模拟矢量引起的一些特殊问题及其解决方案.

ART2 应完成对输入模拟矢量的分类功能，它的基本设计思想仍是采用竞争学习和自稳机制的原理，系统仍由 F_2 和 F_1 两个 STM 层及两者之间的 LTM 层组成. F_2 层的功能是选择竞争优胜者，这一点和 ART1 并没有区别，因此它的设计也没有什么特殊之处. F_1 层的设计是全系统的核心，各种矛盾集中于此，它同时要满足各方面(有时是相互矛盾的)要求. 我们首先罗列这些要求：

(1)F_1 在读进输入矢量和由顶向下的矢量后，应能对这两个模拟矢量进行匹配比较. 如何用神经元完成模拟矢量之间的相似度计算是需要首先解决的问题.

(2)当这两个矢量的相似程度足够高(或者说匹配得足够好)，就应当将它们适当“融合”后送往 LTM 层，该层的相应系数矢量通过学习后将趋向于此融合矢量. 一个重要的问题是使此融合矢量能汇聚输入及由顶向下两个矢量的共同突出特征. 这将使 ART2 的快速学习特点得以实现，也就是说，对于分类而言，每送入一次输入矢量样本就可以将其全部重要特征存入长期记忆之中，而无需反复多次的学习. 另一个需要解决的问题是，在 LTM 系数的学习过程中不能由于这些系数的变化造成优胜者换位的现象，否则学习是不能稳定的. 当输入及由顶向下矢量之间的相似程度不够高时，F_1 应能正确地向 F_2 发出重置信号.

(3)当启用一个未被占据的新输出端时，与该端相应的由顶向下 LTM 系数的初值可以设置为接近于零的值或设置为非零值. 如果是后一种情况，很难恰巧与输入矢量达到良好的匹配，此时免不了发生重置，但是这种重置不允许发生，否则永远不会有新端被启用. 所以，我们只能设这些初值为接近于零，并依据此初值设置来防止对新启用端的重置. 在 ART1 中用增益控制中心 G_1 来完成此任务；而在 ART2 中必须采用其他方法.

(4)在警戒参数选得较高的情况下，两个矢量之间的非本质微小差异也会导致重量的发生，这种错误、重置的现象应该完全消除. 我们可以通过以下几条途径来达到这一点：在各个处理层次中应保持相应的矢量规格化(即使其模等于 1)，应尽量压低或削弱背景噪声(或称为基底噪声)，突出各矢量的重要特征而减弱非重要特征.

以上这几条措施对于提高分类的可靠性也是非常有用的.

为了通盘解决上列诸项问题和存在的矛盾，ART2 采用了一种三层结构，其中包括正反馈、规格化及非线性变换. 如图 5.10 所示，其中 F_1 包含上、中、下三层，在本图中只画出了各层的第 i 个处理单元，图中左侧是调整子系统. 由于其中包含了上、中、下三个层，为了避免混淆，我们以后将分别称为“特征表示场”及“类别表示场”. 我们将结合此图来解释 ART2 的工作原理.

一、F_1 场(STM)中第 j 个处理单元的描述

该系统输入的观察矢量 $\boldsymbol{X}$ 是一个 N 维模拟矢量，它的 N 个分量是 $x_0,x_1,\cdots,x_{N-1}$. 在 F_1 中有相应的 N 个处理单元，每个单元都包括上、中、下三层，图 5.10 中只画出了第 j 个处理单元的结构. 可以看到，每一层里都包括两种功能不同的神经元，一种是空心圆，另一种是实心圆，下面分别介绍其功能.

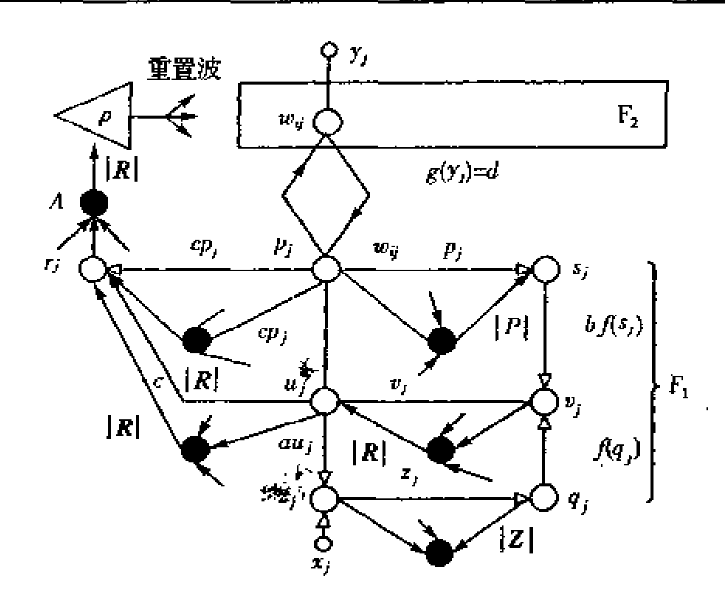


图 5.10　ART2 神经网络的示意图

(1)空心圆. 每个空心圆所代表的神经元可能有两种输入,一种是兴奋激励,用空心箭头表示;另一种是抑制激励,用实心箭头表示. 设神经元的活动电位(也就是它的输出)为 V. 所有兴奋激励的总和为 J^+,所有抑制激励的总和为 J^-. 那么根据神经生理学的研究结果,可以知道变量满足下列微分方程(其中 ϵ 和 A 都是远小于 1 的正实数,且 ϵ 远小于 A,B 远小于 1,C 远小于 D,且 D 接近于 1):

$$\epsilon\frac{\mathrm{d}V}{\mathrm{d}t}=-AV+(1-BV)J^+-(C+DV)J^- \tag{5.3.10}$$

不难看出,当不存在任何激励时(即 $J^+=J^-=0$),V 将趋向于 0,即处于抑制状态. 在式(5.3.10)中,ϵ、B 和 C 这三个参数分别较 A、1 和 D 小得多,因此可以假设 $B=0,C=0,\epsilon\to0$. 根据这一假设,在我们所考虑的时间尺度范围内,可以认为 V 在"瞬间"即能达到其稳定解之值. 这样,可以得到式(5.3.10)的近似解为:

$$V=\frac{J^+}{A+DJ^-} \tag{5.3.11}$$

由于 D 接近于 1,而 A 是一个远小于 1 的正实数,为了简化计算,我们可以用下列近似式来计算一个空心圆神经元的活动电位(即其输出):

$$V=\frac{J^+}{e+J^-} \tag{5.3.12}$$

其中 e 表示一个远小于 1 的正实数.

(2)实心圆. 实心圆神经元的功能是求其输入矢量之模. 例如,在图 5.10 中,与标记为 p_j 的空心圆相连的实心圆,除了 p_j 以外还与 $p_0,p_1,\cdots,p_{N-1}$ 等各空心圆相连. 设 $\boldsymbol{P}=[p_0p_1\cdots p_{N-1}]$,那么此实心圆的输出即等于 $\boldsymbol{P}$ 的模,换言之,它的输出可表示为

$$|\boldsymbol{P}|=\{p_0^2+p_1^2+\cdots+p_{N-1}^2\}^{1/2} \tag{5.3.13}$$

如对普通神经元的各输入施以平方型非线性变换并对其输出施以开方型非线性变换,就可以完成这一运算功能. 类似于此,与 v_j、z_j、u_j 和 cp_j 等相连的实心圆,其输出分别等于 $\|\boldsymbol{V}\|$、$\|\boldsymbol{Z}\|$、$\|\boldsymbol{U}\|$ 和 $c\|\boldsymbol{P}\|$. 由图 5.10 可以看到,各实心圆送出的都是抑制激励.

在图 5.10 中,F_1 的第一层(底层)和第二层(中层)构成一个闭合的正反馈回路,其中标记为 z_j 的神经元接受输入信号 x_j,而标记为 v_j 的神经元接受上层送来的信号 $bf(s_j)$. 这个回路中还包括两

次规格化运算和一次非线性变换.底层的输入方程和规格化运算可以表示如下：

$$z_j = x_j + au_j \tag{5.3.14}$$

$$q_j = \frac{z_j}{e + |\boldsymbol{Z}|} \tag{5.3.15}$$

其中 e 是一个很小的正实数，相对于 $|\boldsymbol{Z}|$ 可忽略不计. 中层的方程和规格化运算可表示为

$$v_j = f(q_j) + bf(s_j) \tag{5.3.16}$$

$$u_j = \frac{v_j}{e + |\boldsymbol{V}|} \tag{5.3.17}$$

其中 e 相对于 $|\boldsymbol{V}|$ 而言要小得多，仍可忽略不计. 在底层至中层以及上层至中层之间，对传送的信号进行了非线性变换，此变换用函数 $f(x)$ 来表示，在具体实现时，$f(x)$ 可以取如下两种形式：

第一种形式：

$$f(x) = \begin{cases} \dfrac{2\theta x^2}{(x^2 + \theta^2)}, & 0 \leqslant x \leqslant \theta \\ x, & x > \theta \end{cases} \tag{5.3.18}$$

第二种形式：

$$f(x) = \begin{cases} 0, & 0 \leqslant x \leqslant \theta \\ x, & x > \theta \end{cases} \tag{5.3.19}$$

这两种非线性函数的作用都是对小幅度信号进行抑制，其区别在于前者是连续函数而后者是非连续的，后者在实现时更方便一些. 以上各式中的一些参数 a，b 和 θ 是通过大量实验来确定的.

F_1 的第二层（中层）和第三层（上层）构成另一个闭合正反馈回路，其中标记为 p_j 的神经元既接受中层送来的信号 u_j 又接受 F_2 场送来的信号. 在这个回路中也包含两次规格化运算和一次非线性变换. 在上层进行的规格化运算可表示为

$$s_j = \frac{p_j}{e + |\boldsymbol{P}|} \tag{5.3.20}$$

其中 e 仍可忽略不计. 上中层之间的非线性变换已如前述. p_j 可表示为：

$$p_j = u_j + \sum_{i=0}^{M-1} g(y_i) w'_{ji} \tag{5.3.21}$$

上式右侧第二项表示 F_2 场对 p_j 神经元的输入，w'_{ji} 是由顶向下的 LTM 系数，$g(y_i)$ 将在下面说明.

二、F_2 场中所完成运算的描述

跟据 F_1 场上层给出的矢量 $\boldsymbol{P} = [p_0 p_1 \cdots p_{N-1}]$ 以及由 F_1 至 F_2 的 LTM 权重系数 w_{ij} ($j = 0 \sim (N-1)$，$i = 0 \sim (M-1)$) 可以算出 F_2 场（STM）的输入矢量，其中 $t_i = \sum_{j=0}^{N-1} w_{ij} p_j$. 按照竞争学习机制，$F_2$ 场的输出矢量 $\boldsymbol{Y} = [y_0 y_1 \cdots y_{N-1}]$ 由下式决定：

若 $t_I = \max\{t_i, i = 0 \sim (M-1)\}$，则 $y_I = 1$ 且 $y_i = 0$，当 $i \neq I$

如果我们选择式（5.3.21）中的变换函数为 $g(y_i) = dy_i$，其中 d 是一个实常数，那么可用下式计算 p_j，

$$p_j = u_j + dw'_{jI}, j = 0 \sim (N-1)（因为当 i \neq I, y_i = 0 时, y_I = 1） \tag{5.3.22}$$

尚未受到激励时，所有 $y_i=0$，这时 $p_j=u_j$.

三、F_1 场和 F_2 场之间 LTM 权重系数的学习

当 $F_2 \to F_1$ 的由顶向下矢量和输入观察矢量的相似度足够高或开辟了一个新输出端，则进入系数 LTM 的学习阶段. 学习公式是：

由顶向下($F_2 \to F_1$)：

$$w'_{ji}(k+1) = w'_{ji}(k) + g[y_i(k)\{p_j(k) - w'_{ji}(k)\}]$$
$$i = 0 \sim (M-1), j = 0 \sim (N-1) \qquad (5.3.23)$$

由底向上($F_2 \to F_1$)：

$$w_{ij}(k+1) = w_{ij}(k) + g[y_i(k)\{p_j(k) - w_{ij}(k)\}]$$
$$i = 0 \sim (M-1), j = 0 \sim (N-1) \qquad (5.3.24)$$

如果 I 场中选出的优胜输出端是 I，那么 $g(y_I)=d$，当 $i \neq I$ 时，$g(y_i)=0$. 这样式(5.3.23)及(5.3.24)可表示为

$$\begin{cases} w'_{jI}(k+1) = w'_{jI}(k) + d\{p_j(k) + w'_{jI}(k)\}, \\ w'_{ji}(k+1) = w'_{ji}(k), \text{当 } i \neq I, \end{cases} \quad j = 0 \sim (N-1) \qquad (5.3.25)$$

$$\begin{cases} w_{Ij}(k+1) = w_{Ij}(k) + d\{p_j(k) - w_{Ij}(k)\}, \\ w_{ij}(k+1) = w_{ij}(k), \text{当 } i \neq I, \end{cases} \quad j = 0 \sim (N-1) \qquad (5.3.26)$$

将式(5.3.22)代入式(5.3.25)和(5.3.26)，就能得到

$$w'_{jI}(k+1) = w'_{jI}(k) + d(1-d)\left\{\frac{u_j(k)}{(1-d)} - w'_{jI}(k)\right\}, j = 0 \sim (N-1) \quad (5.3.27)$$

$$w_{Ij}(k+1) = w_{Ij}(k) + d(1-d)\left\{\frac{u_j(k)}{(1-d)} - w_{Ij}(k)\right\}, j = 0 \sim (N-1) \quad (5.3.28)$$

其中 I 是竞争优胜输出端或新开辟输出端的编号. 对于参数 d，应指出它有别于一般学习公式中的步幅参数. 因为在 ART2 中采用的是"快速学习"方式，即每呈现一个输入矢量，系数"立即"进入其平衡状态，无需像普通的慢速学习方式那样要进行很多步调整才能进入平衡态. 对于后者，步幅应选得足够小，而参数 d 的选择则取决于快速学习的其他制约因素.

至此为止，我们对于正场中的上、中、下三层及由其构成的两个正反馈而路，对于 F_1 场和 F_2 场之间的相互作用及场间 LTM 系数的学习调整等问题已介绍清楚并给出了有关的计算公式. 在进一步介绍调整子系统前，需讨论这些结构中所用的一些矢量的物理意义，其中最重要的是 $\boldsymbol{P}$、$\boldsymbol{Q}$、$\boldsymbol{W}_I$、$\boldsymbol{W}'_I$ 等几个矢量. 当 F_2 场尚未受到激励而 F_1 场只受到输入矢量 $\boldsymbol{X}$ 作用时，可以看到，在两个正反馈回路中的联合作用下所产生的矢量 $\boldsymbol{U}$ 的模为 1，它反映了输入矢量 $\boldsymbol{X}$ 的重要特征并抑制了其中的基底噪声. 由于 F_2 未受激励(即 $\boldsymbol{Y}=0$)，由顶向下传送的矢量等于零，根据式(5.3.11)，由底向上传送的矢量 $\boldsymbol{P}$ 即等于 $\boldsymbol{U}$. 因此，我们可以得到如下结论. 当 F_2 未受激励或者启用一个新输出端而其相应的由顶向下 LTM 系数皆取初值为零时，F_1 场中的各矢量满足如下关系：

(1)设 $\boldsymbol{S}=[s_0 s_1 \cdots s_{N-1}]$，则有 $\boldsymbol{S}=\boldsymbol{P}=\boldsymbol{U}=\boldsymbol{V}/|\boldsymbol{V}|$，且 $|\boldsymbol{S}|=|\boldsymbol{P}|=|\boldsymbol{U}|=1$.

(2)$\boldsymbol{U}$ 反映了 $\boldsymbol{X}$ 的重要特征并抑制了它的基底噪声.

(3)$\boldsymbol{Z}=\boldsymbol{X}+a\boldsymbol{U}$

(4)设 $\boldsymbol{Q}=[q_0 q_1 \cdots q_{N-1}]$，则 $\boldsymbol{Q}=\boldsymbol{Z}/|\boldsymbol{Z}|$，且 $|\boldsymbol{Q}|=1$.

当 F_2 受到激励并选出了一个已被占用过的输出端 I 时，按照式(5.3.21)和式(5.3.22)，

由顶向下传送的矢量为 dW_I'，$W_I=[w_{0I}'w_{1I}'\cdots w_{N-1,I}']$，而矢量 $\boldsymbol{P}$ 可表示为 $\boldsymbol{P}=\boldsymbol{U}+dW_I'$. 可以看到，在两个正反馈回路的作用下，$\boldsymbol{U}$ 融合了 $\boldsymbol{X}$ 和 W_I' 的重要特征且抑制了 $\boldsymbol{X}$ 的基底噪声. 值得注意，当 $\boldsymbol{X}$ 和 W_I' 足够相似时，$\boldsymbol{P}$ 和 $\boldsymbol{U}$ 也是足够相似的；反之，当 $\boldsymbol{X}$ 和 W_I' 不相似时，$\boldsymbol{P}$ 和 $\boldsymbol{U}$ 之间的差异也比较大. 另外，设 $W_I=[w_{0I}'w_{1I}'\cdots w_{N-1,I}']$，取其与 $\boldsymbol{P}$ 之点积就得到 t_I. 现在就可以用一个调整子系统来计算 $\boldsymbol{P}$ 和 $\boldsymbol{U}$ 的相似度，当此相似度低于某个警戒参数时就应该对 F_2 重置；反之，就进入 LTM 参数学习阶段. 之所以选择 $\boldsymbol{P}$ 和 $\boldsymbol{U}$ 来比较相似度，而不直接比较 $\boldsymbol{X}$ 和 W_I'，除了因为在启用新端时前者仍保持完全一致而后者差异极大（因而前者不会引起重置而后者却会造成重置）以外，$\boldsymbol{P}$ 和 $\boldsymbol{U}$ 比较的是去除了基底噪声的主要特征，而 $\boldsymbol{X}$ 和 W_I' 的比较中还包含了非主要特征和基底噪声，当警戒参数设置得较高时，这就使得一些次要特征和因素也会引起不应有的重置. 另外还要强调指出，如果 LTM 系数进入了学习阶段，这时由顶向下的矢量 W_I' 是在变化的，因而 $\boldsymbol{P}$ 也处于变化之中. 我们必须保证，如果在调整开始时未发生重置，那么在其后的全部学习阶段都不能发生重置，否则就不能进行稳定的学习.

四、调整子系统的工作原理以及 W_{ji}' 和 W_{ij} 初值的选择，参数 d 和参数 c 的选择

图 5.10 左侧是调整子系统的示意结构，其中标志为 r_j 的空心圆神经元是第 j 个处理单元的一个组成部分. 根据式(5.3.13)，其输出可用下式计算：

$$r_j=\frac{u_j+cp_j}{e+|\boldsymbol{U}|+c|\boldsymbol{P}|},j=0\sim(N-1)\tag{5.3.29}$$

由于 e 远小于 1，在下面的计算中可略去. 实心圆 A 的输出级等于矢量 $\boldsymbol{R}$ 之模，$\boldsymbol{R}=[r_0, r_1,\cdots,r_{N-1}]$，这可以表示为

$$|\boldsymbol{R}|=\left[\sum_{j=0}^{N-1}r_j^2\right]^{\frac{1}{2}}\tag{5.3.30}$$

显而易见，$\boldsymbol{U}$ 与 $\boldsymbol{P}$ 的相似度越高，则 $|\boldsymbol{R}|$ 越接近于 1. 这样，我们可以选择一个警戒参数 ρ，$0<\rho<1$，当 $|\boldsymbol{R}|>\rho$ 时，无需重置；反之，则需要对 F_2 进行重置. 为了探讨当新启用一个未占用过的输出端时或者在 LTM 系数的调整过程中是否有可能发生不该有的错误重置，我们应仔细研究在各种情况下 $|\boldsymbol{R}|$ 是如何变化的. 如果在式(5.3.29)中令 $e=0$ 且已知 $|\boldsymbol{U}|\approx1$，那么式(5.3.29)和式(5.3.30)可以改写为如下形式：

$$\boldsymbol{R}=\frac{\boldsymbol{U}+c\boldsymbol{P}}{1+c|\boldsymbol{P}|}\tag{5.3.31}$$

$$|\boldsymbol{R}|=\frac{[1+2c|\boldsymbol{P}|\cos(\boldsymbol{U},\boldsymbol{P})+c^2|\boldsymbol{P}|^2]^{\frac{1}{2}}}{1+c|\boldsymbol{P}|}\tag{5.3.32}$$

其中 $\cos(\boldsymbol{U},\boldsymbol{P})$ 表示矢量 $\boldsymbol{U}$ 和 $\boldsymbol{P}$ 之间的夹角的余弦值. 我们已知 $\boldsymbol{P}=\boldsymbol{U}+dW_I'$（见式(5.3.22)），因此 $\boldsymbol{U}$ 和 $\boldsymbol{P}$ 的夹角的余弦可用下式计算：

$$\cos(\boldsymbol{U},\boldsymbol{P})=\frac{\boldsymbol{P}\cdot\boldsymbol{U}}{|\boldsymbol{P}||\boldsymbol{U}|}=\frac{|\boldsymbol{U}|^2+d\boldsymbol{U}\cdot W_I'}{|\boldsymbol{P}||\boldsymbol{U}|}\tag{5.3.34}$$

其中 $\boldsymbol{P}\cdot\boldsymbol{U}$ 和 $\boldsymbol{U}\cdot W_I'$ 表示两个矢量的点积（内积）. 如注意到 $|\boldsymbol{U}|\approx1$，并且再次利用两个矢量的点积及其夹角余弦值之间的关系，就能够得到

$$|\boldsymbol{P}|\cos(\boldsymbol{U},\boldsymbol{P})=1+d|W_I'|\cos(\boldsymbol{U},W_I')\tag{5.3.35}$$

$$|\boldsymbol{P}|=[1+2d|W_I'|\cos(\boldsymbol{U},W_I')+d^2|W_I'|^2]^{\frac{1}{2}}\tag{5.3.36}$$

将式(5.3.32)和(5.3.36)综合，即可得到

$$|\boldsymbol{R}|=\frac{[(1+c)^2+2(1+c)\,|\,cd\boldsymbol{W}_I'|\cos(\boldsymbol{U},\boldsymbol{W}_I')+|\,cd\boldsymbol{W}_I'|^2]^{\frac{1}{2}}}{1+[c^2+2c\,|\,cd\boldsymbol{W}_I'|\cos(\boldsymbol{U},\boldsymbol{W}_I')+|\,cd\boldsymbol{W}_I'|^2]^{\frac{1}{2}}} \tag{5.3.37}$$

此式中的 $\cos(\boldsymbol{U},\boldsymbol{W}_I')$ 表示的是由底向上矢量和由顶向下矢量之间的相似度. 式中的 c 和 d 是两个可供设计者选择的常参数. $|\boldsymbol{W}_I'|$ 是由顶向下矢量之模.

五、关于参数 θ、a 和 b 的选择

我们在前面曾指出这几个参数需通过实验来加以确定. 首先，对于 θ 而言，一个初步的选择是可以令 $\theta=1/\sqrt{N}$. 在此条件下，如果输入矢量是一个各分量取值皆相等的均匀矢量，那么通过 F_1 场的中层及下层正反馈回路的作用，由于非线性变换的结果，矢量 $\boldsymbol{U}$ 的所有分量都压到零值，因而 F_1 场不会向 F_2 场送出任何信号. 换言之，这时可将均匀矢量当做一个噪声矢量. 显然，当选 θ 值超过 $1/\sqrt{N}$ 时，基底噪声抑制和对比度增强的效果都更加提高，但是有用信号也被切除得更多.

实验证明，参数 a 和 b 选得很小且 ρ 也很小时，分类结构不能稳定. 此外，如果选 $\theta=1/\sqrt{N}$，$a=b=10$，$c=0.1$，$d=0.9$ 那么在 ρ 选各种数值的情况下都能获得稳定的分类结果.

六、ART2 网络的学习算法

①初始化. 给参数 θ,a,b,c,ρ 和 d 赋予相应的初值. 自下面上的权向量 $\boldsymbol{W}$ 和自上而下的权向量 $\boldsymbol{W}'$ 赋初值.

②给网络输入模式 $\boldsymbol{X}=(x_0,x_1,\cdots,x_{N-1})$，$x\in[0,1]$.

③根据式(5.3.14)、(5.3.15)、(5.3.16)、(5.3.17)、(5.3.19)、(5.3.20)和(5.3.21)；计算 F_1 场中的各矢量：$\boldsymbol{Z},\boldsymbol{Q},\boldsymbol{V},\boldsymbol{U},\boldsymbol{S}$ 和 $\boldsymbol{P}$.

④F_2 场中输入矢量 $\boldsymbol{T}=[t_0,t_1,\cdots,t_{M-1}]$，其中 $t_i=\sum_{j=0}^{N-1}w_{ij}p_j$，$i=0\sim(M-1)$，若有 $t_I=\max\{t_i,i=0\sim(M-1)\}$，则 $y_I=1$ 且 $y_i=0$，当 $i\neq I$，则 I 节点为获胜节点. 当场 F_2 未受激励时，所有 $y_i=0$.

⑤信息反送. 由 F_2 场的获胜节点 I 送回自上而下的权向量 $\boldsymbol{W}_I'$，根据式(5.3.29)、(5.3.30)计算出 $|\boldsymbol{R}|$ 值.

⑥警戒线检测. 若 $|\boldsymbol{R}|>\rho$，则接受 I 为获胜节点. 转⑦. 否则发 Reset 信号，置 $I=0$(不允许其再参加竞争)，开始搜索阶段，转③.

⑦根据式(5.3.27)和式(5.3.28)调整自下面上及自上而下的权向量，使其以后对与 $\boldsymbol{X}$ 相似的输入更容易获胜，且具有更大的相似性.

5.3.5 ART 神经网络在人像识别中的应用

一、ART1 在人像识别中的应用

自适应人像识别神经网络根据以上改进的 ART1 神经网络所设计，其输入模式采用小波变换方法对数据进行特征抽取并编码而得到，其输出表示类别. 如图 5.11 所示，它包括五层和六个控制节点，其功能如下：

第一层：输入层，输入信号 $\boldsymbol{X}=\{x_0,x_1,\cdots,x_n\}$ 为 n 维向量，其输出信号 $\boldsymbol{O}_1=\boldsymbol{X}$. $\boldsymbol{O}_2$ 为由“2/3 规则”产生的信号，记为 n 维向量 $\boldsymbol{S}^*$，$\boldsymbol{S}^*$ 为记忆模式与输入模式的编合模式：

$$\boldsymbol{S}^*=\boldsymbol{Z}\times\boldsymbol{X} \tag{5.3.38}$$

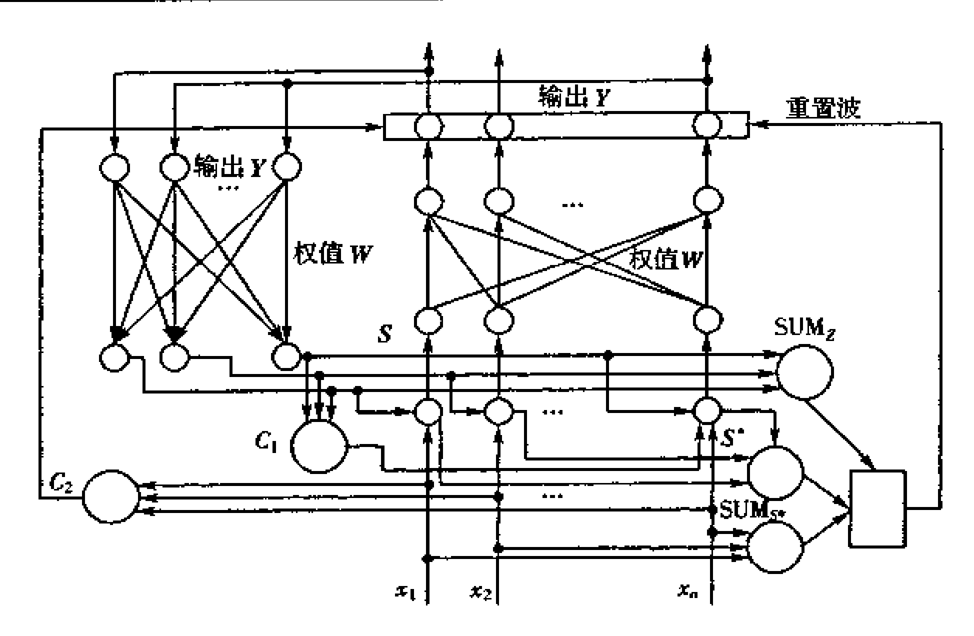


图 5.11　ART1 人像识别神经网络

Z 在后面介绍

第二层：归一化层，输入为第一层输出 O_1，输出为 n 维向量 $\boldsymbol{S}$. 该层的作用是使输入矢量规格化：

$$s_j = \frac{x_j}{\left(\sum_{i=0}^{n-1} x_i^2\right)^{1/2}} \tag{5.3.39}$$

第二层和第三层之间为自下而上的全互连前向网络，其连接权矩阵为 $\boldsymbol{W}$，第三层的输入为 $\boldsymbol{S}$，输出为 m 维向量 $\boldsymbol{T}$：

$$t_i = \sum_{j=0}^{n-1} w_{ij} s_j, \qquad i = 0,1,\cdots,m-1 \tag{5.3.40}$$

第四层：竞争网络层. 输入为 $\boldsymbol{T}$，输出为 m 维向量 $\boldsymbol{Y}$，通过该层的竞争，产生出胜者，只有获胜者输出为 1，其他为 0：

$$t_I = \max\{t_i, i = 0,1,\cdots,m-1\} \tag{5.3.41}$$

$$y_k = \begin{cases} 1, & k = I \\ 0, & k \neq I \end{cases} \tag{5.3.42}$$

第四层到第五层为自上而下的反馈层，其连接权矩阵为 $\boldsymbol{W}'$，第五层的输入为 $\boldsymbol{Y}$，输出为 n 维向量 $\boldsymbol{Z}$：

$$z_j = \sum_{k=0}^{m-1} w_{kj} y_k \tag{5.3.43}$$

因为 $\boldsymbol{Y}$ 只有获胜者输出为 1，其他为 0，所以：

$$z_j = w_{Ij} \tag{5.3.44}$$

n 维向量 $\boldsymbol{Z}$ 为该获胜模式所记忆的向量；

控制节点 G_1：获胜记忆模式 $\boldsymbol{Z}$ 的“或非”. 即只有当 $\boldsymbol{Z}$ 全为 0 时，$G_1=1$，其他情况为 0；

控制节点 G_2：输入模式 $\boldsymbol{X}$ 的“或”. 即只有当 $\boldsymbol{X}$ 全为 0 时，$G_2=0$，其他情况为 1；

控制节点 SUM$_Z$：计算获胜记忆模式的非零分量的个数.

$$\mathrm{SUM}_Z = \| \boldsymbol{Z} \| = \sum_{i=1}^{n} w_k \tag{5.3.45}$$

控制节点 SUM_{S^*}：计算综合模式 $\boldsymbol{S}^*$ 的非零分量的个数.

$$\mathrm{SUM}_S^* = \sum_{i=1}^{n} w_B x_i \tag{5.3.46}$$

控制节点 SUM_X：计算输入模式 $\boldsymbol{X}$ 的非零分量的个数.

$$\mathrm{SUM}_X = \sum_{i=1}^{n} x_i \tag{5.3.47}$$

控制节点 Reset：输入为 SUM_Z、SUM_S、SUM_X、输出为 $\boldsymbol{R}$：

$$\boldsymbol{R} = \begin{cases} 0, & f(\mathrm{SUM}_{S^*}/\mathrm{SUM}_Z) * (\mathrm{SUM}_{S^*}/\mathrm{SUM}_X) > \rho \\ 1, & \text{否则} \end{cases} \tag{5.3.48}$$

函数 f 如公式(5.3.37)定义，当 $\boldsymbol{R}=1$ 时，Reset 向第四层发重置波.

网络训练及识别算法(FER)：

①初始化输入模式 $\boldsymbol{X}=0$、输出模式 $\boldsymbol{Y}=0$、连接权值 $\boldsymbol{W}$、$\boldsymbol{W}'$、控制信号 $G_1=1$、$G_2=0$、Reset $=0$，已训练类别数为 premode，网络容量为 summode.

②输入一新模式 $\boldsymbol{X}_i=\{x_0, x_1, \cdots, x_N\}$，Reset$=0$，计算 G_2.

③如果 $G_2=0$，则转②.

④由公式(5.3.39)和(5.3.40)计算 $\boldsymbol{S}$、$\boldsymbol{T}$.

⑤由公式(5.3.41)～(5.3.44)计算 I、$\boldsymbol{Y}$、$\boldsymbol{Z}$，计算 G_1.

⑥如果 $G_1=1$，则转②.

⑦由公式(5.3.38)、(5.3.45)～(5.3.48)计算 $\boldsymbol{S}^*$、SUM_Z、SUM_X、SUM_{S^*} 和 Reset.

⑧如果 Reset$=0$，则转⑬.

⑨搜索记数 NUM+1，置 y_I、t_I 为 0.

⑩NUM 小于等于 premode，则进入搜索阶段，转⑤.

⑪增加新类别 premode+1，连接权值 $\boldsymbol{W}$，premode，j，$0 \leqslant j \leqslant N$，赋初值.

⑫由公式(5.3.40)重新计算 $\boldsymbol{T}$，转⑤.

⑬由公式(5.3.34)和(5.3.35)调整连接权值 $\boldsymbol{W}_j$、$\boldsymbol{W}'$，NUM$=0$，转②.

改进的 ART1 学习算法不仅考虑了综合模式与输入模式的相似度，而且考虑了综合模式与记忆模式的相似度，使得该算法的识别误差大大地减小. 另外，采用小波变换为特征抽取工具，并结合 PART1 神经网络所进行的人像识别具有小波变换和 ART 的双重优点，其识别的自适应性、识别速度、网络容量的可扩充性和稳定的识别率都是很好的. 对于已学习过的样本其识别率可高达 100%，其识别速度和其自适应性是 BP 网和遗传算法所无法比拟的.

二、使用连续型 ART2 网络进行人像识别

自适应人像识别神经网络是根据以上的连续型神经网络 ART2 设计而成的，其输入模式包括对特征图像降维处理之后，所得到的数据信息，以及提取出的眼睛、鼻子和嘴等的几何比例特征两部分. 主要过程是：根据 Laplace 边缘检测所得到的图像，按照前面所给出的瞳孔定位算法，定位瞳孔所在的位置，然后根据瞳孔位置重新精确定位眼睛，并根据眼睛的位置，对人像进行倾斜和旋转的补偿算法；同时为了克服抬头(或低头)的影响，对每个待识别的人像增加

以眼睛为中心,垂直于图像平面方向的旋转±5°、±10°的图像,这样就可以得到五幅图片.然后在补偿后的人像上,根据鼻子、嘴的定位算法,精确定位鼻子和嘴的位置,并找到鼻孔和嘴角的位置,根据瞳孔、鼻孔和嘴角的位置,得到三者之间的连线的距离、面积、比例等特征,并且对这些特征的数值进行变换,使其更适应于 ART2 的输入.然后根据所定位的面部特征,分离出整个图像中的包含眼睛、鼻子、嘴的特征区域,并对特征区域的图像进行尺度和灰度的标准化处理.对标准化后所得到的图像部分先进行一次二维小波变换,整个特征区域被分成四个部分,把四个部分的对应像素值都加起来,这样就能保留低频部分的基本特征,还能保留图像的高频部分的边缘特征.叠加之后,所得到的向量的维数为原来特征区域的维数的 1/4.然后利用主分量分析的方法,进一步对特征区域降维,把影响较小的部分舍弃,使得输入向量的维数可以进一步地减少.然后把经过处理的值域在[0,1]之间的数作为网络的输入.这样,ART2 的输入部分已经确定,其输出部分表示类别.具体的实现过程如下:

ART2 网络训练及用于人像识别的算法:

①初始化输入模式 $\boldsymbol{X}=0$、输出模式 $\boldsymbol{Y}=0$、连接权值 $\boldsymbol{W}$、$\boldsymbol{W}'$ 及重置信号 Reset=0,已训练类别数为 Premode,网络的容量为 Summode,设置各参数的初值:$\theta=1/\sqrt{N}$,$a=b=10$,$c=0.1$,$d=0.9$ 和 $\rho=0.8$.

②输入一新模式 $\boldsymbol{X}=\{x_0,x_1,\cdots,x_{N-1}\}$.

③根据式(5.3.14)、(5.3.15)、(5.3.16)、(5.3.17)、(5.3.19)、(5.3.20)和(5.3.21);计算 F_1 场中的各矢量:$\boldsymbol{Z},\boldsymbol{Q},\boldsymbol{V},\boldsymbol{U},\boldsymbol{S}$ 和 $\boldsymbol{P}$.

④F_2 场中输入矢量 $\boldsymbol{T}=[t_0 t_1\cdots t_{M-1}]$,其中 $t_i=\sum\limits_{j=0}^{N-1} w_{ij}p_j$,$i=0\sim(M-1)$,若有 $t_I=\max\{t_i,i=0\sim(M-1)\}$,则 $y_I=1$ 且 $y_i=0$,当 $i\neq I$,则 I 节点为获胜节点.当场 F_2 未受激励时,所有 $y_i=0$.

⑤信息反送.由 F_2 场的获胜节点 I 送回自上而下的权向量 $\boldsymbol{W}'_I$,根据式(5.3.29)、(5.3.30)计算出 $|\boldsymbol{R}|$ 值.并进行警戒线检测,设置 Reset 值.

⑥如果 Reset=0,则转⑨;否则搜索计数 Num+1,置 $y_I=0$,$t_I=0$.

⑦若 Num≤Premode,则进入搜索阶段,转③.

⑧增加新类别 Premode+1,对连接权值 $\boldsymbol{W}_{\text{premode}}$、$\boldsymbol{W}'_{\text{premode}}$ 赋初值;转③.

⑨由公式(5.3.27)、(5.3.28)调整连接权值 $\boldsymbol{W}_I$,$\boldsymbol{W}'_I$,置 Num=0,转②.

§5.4 神经认知机

认知机及神经认知机是由日本学者 Fukishima(福岛邦邦彦)提出的,它是一具有自组织功能的多层神经网络模型,用于结构化的字符识别.这种模型虽然不便于电子电路化,但却便于计算机模拟,也常用于视觉和听觉信息处理系统.

神经认知机对可塑性突触的形成作如下假设:

(1)从神经元 x 到神经元 y 的突触连接,只有在神经元 x 处于激活的情况下才被强化.

(2)如果在神经元 y 的近旁存在有比 y 更强的激活神经元 y',则从 x 至 y 的突触连接就不进行强化.这也就是说,这种突触连接的强化应符合"最大值检出假说",即在某一小区域(称之为邻域)内存在的一神经元集合中,只有输出最大的神经元才发生输入突触的强化.这样,对

某一邻域内的神经元，其输入突触都可能被强化. 但根据假说之二，却只限定其中的一个神经元会发生强化. 根据这一假说，当其中某一神经元受到损伤时，其邻近的其他神经元将会取代它，起到自动修复的作用. 这一特性与神经细胞的实际功能是相符合的.

基于上述假设即有图 5.12 所示的多层神经网络模型. 模型采用分流抑制结构. 神经元的输入和输出是与激活脉冲密度(短时间平均)的非负模拟成比例.

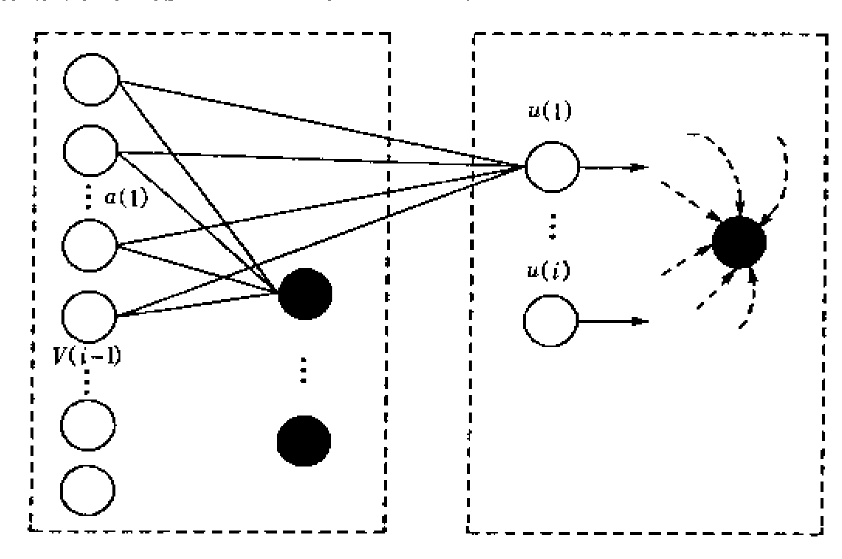


图 5.12 多层神经网络

设神经元的兴奋性突触的输入为 $U(1), U(2), \cdots U(N)$ 和抑制性突触的输入为 $V(1)$, $V(2), \cdots, V(M)$，则神经元的输出为：

$$V = \varphi\left[\frac{1 + \sum_{i=1}^{N} a(i)U(i)}{1 + \sum_{i=1}^{M} b(i)V(i)}\right] \tag{5.4.1}$$

其中

$$\varphi(x) = \begin{cases} x, & x \geqslant 0 \\ 0, & x < 0 \end{cases} \tag{5.4.2}$$

兴奋性突触的结合强度 $a(i)$ 和抑制性结合强度 $b(i)$ 均为非负模拟值. 式中的分子表示神经元的膜电位(短时间平均值). 兴奋性突触的输入将使膜电位上升，而抑制性突触的输入对膜电位起分流作用. 分母则表示某个心膜电位.

如将加到神经元兴奋输入的和记为 e，而将抑制性输入的和写成 h，则有

$$V = \varphi\left[\frac{1+e}{1-h} - 1\right] = \varphi\left[\frac{e+h}{1-h}\right] \tag{5.4.3}$$

$$e = \sum_{i=1} a(i)U(i) \tag{5.4.4}$$

$$h = bV$$

神经认知机的等效框图如图 5.13 所示.

当抑制性输入很小，即 $h \ll 1$ 时，神经元输出决定于 $(e-h)$，即 $V \approx \varphi(e-h)$. 这时认知机与模拟阈值元件特性相一致；当兴奋性和抑制性都很大，即 $e \gg 1, h \gg 1$ 时，神经元的输出将由 R/h 决定；如将兴奋性输入的和 e 及抑制性输入的和 h 写成 $e = \varepsilon x, h = \eta x$，则当 $\varepsilon <$

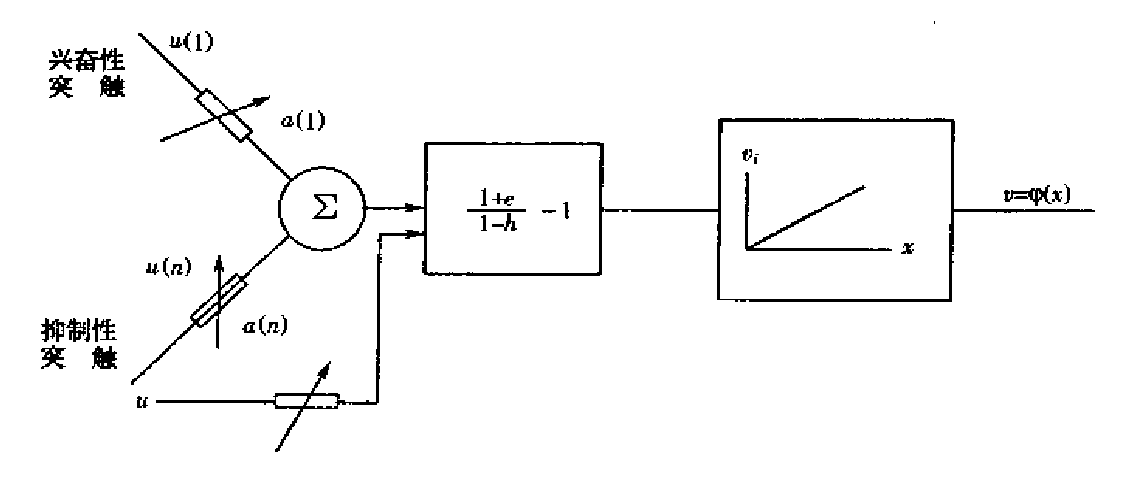


图 5.13 神经认知机的等效框图

η时有

$$V=\frac{(\varepsilon-\eta)x}{1+\eta x}=\frac{\varepsilon-\eta}{2\eta}\left[1+\tanh\left(\frac{1}{2}\lg\frac{1}{x}\right)\right] \tag{5.4.5}$$

即接近于S形饱和特性，与神经生物学感观系统的I/O非线性特性一致. 认知机可以通过自组织获得各种空间模式的识别功能，常用于文字与语言的识别等研究领域.

参考文献

[1]张立明. 人工神经网络的模型及其应用. 上海:复旦大学出版社,1992 年.

[2]袁曾任. 人工神经元网络及其应用. 北京:清华大学出版社,1999 年.

[3]焦李成. 神经网络系统理论. 西安:西安电子科技大学出版社,1990 年.

[4]中野馨. ニューロコンピュタ. 东京:技术评论社,1989 年.

[5]麻生英树. ニューラルネットワーク. 东京:产业图书,1990 年.

[6]Liang Yanchun, Zhou Chunguang. Advances in identification of nonlinear characteristics of packaging based on computational intelligence. Mechanics Research Communication. JAN—FEB. 2000. Vol. 27, No. 1, pp. 15~20

[7]Liang Yanchun, Yang Xiaowei, Zhou Chunguang, Wang Zaishen. Application of Neural Networks to Identification of Nonlinear Characteristics in Cushioning Packaging. Mechanics Research Communication. 1996, Vol. 23, No. 6, pp. 607~613

[8]Liang Y C, Zhou C G, Wang Z S. Identification of Restoring Forces in Non—linear Vibration Systems Based on Neural Networks. Journal of Sound and Vibration. 1997, Vol. 206, No. 1, pp. 103~108

[9]梁艳春,周春光. 基于神经网络方法的包装件非线性特性识别的研究. 力学学报,1997, Vol. 29, No. 4, pp. 496~500

[10]梁艳春,周春光,王在申. 人工神经网络应用地下洞室围岩参数识别的研究. 模式识别和人工智能. 1996, Vol. 9, No. 1, pp71~77

[11]周春光,张冰,程彦丰,胡成全. 遗传算法在训练前向神经网络中的应用. 小型微型计算机系统. 1996, Vol. 17, No. 9, pp. 54—58

[12]周春光,常迪,张兵,权炜,梁艳春. 一种改进的 ART1 算法及其在人像识别中的应用. 小型微型计算机系统, Oct. 1999, Vol. 20, No. 10. , pp751~755

[13]周春光,胜部昭明. 神经网络味觉信号的学习和识别. 吉林大学自然科学学报,1994, No. 1, pp. 31~34

[14]周春光. 味觉信号的特征抽取. 吉林大学自然科学学报,1994, No. 2, pp. 25~28

[15]董哲,郭东伟,周春光. BP 算法样本特性及参数 α,β 两阶段动态调整. 吉林大学自然科学学报,1995, No. 1, pp. 33~37

[16]胡成全,周春光,王艳,吕慧英. 基于神经元网络的广域网路由选择算法,吉林大学自然科学学报. 特刊, Suppl. 1998, pp. 76~80

[17]常迪,周春光,刘小华,梁艳春,徐杰. 人像识别中的特征抽取. 吉林大学自然科学学报, 2000, No. 2, pp. 37~40

[18]梁艳春,杨晓伟,周春光. 神经网络方法在包装件缓冲层非线性特性识别中的应用. 包装工程, 1996, Vol. 17, No. 2, pp. 16~20

[19]梁艳春,杨晓伟,李爱杨,周春光. 采用预处理的心电信号数据压缩的神经网络方法. 生物

医学工程，1996，Vol. 25，No. 3，pp. 257～261

[20]梁艳春，周春光. 基于人工神经网络的非线性振动系统恢复力识别的研究. 吉林工业大学学报，1995，Vol. 25，Supplement，pp. 8～12

[21]Liang Yanchun，Zhou Chunguang，Wang Zhiliang. Advance in Identification Characteristics of Packaging Based on Computational Intelligence. 11th IAPARI World Conference on Packaging，July 1999，Singapore，pp458－463. The Westin Stamford and Westin Plaza Hotel.

[22]Zhou Chunguang，Zheng Yan. A Stusy of GAS Idetification Using Neural Networks. Nanjing China：IEEE International Conference on Neural Networks and Signal Processing，Dec. 1995，Conference Proceeding Volume I，pp. 457～460

[23]Wang Yan，Zhou Chunguang. Another Adaptive Routing Algorithm in Wide Area Network Based on Neural Network. Nanjing China：IEEE International Conference on Neural Networks and Signal Processing，Dec. 1995，Conference Proceeding Volume I，pp. 409～412

[24]Lian Yanchun，Zhou Chunguang. Application of Neural Networks in Parameter Identification in Rock Engineering. Nanjing China：IEEE International Conference on Neural Networks and Signal Processing，Dec. 1995，Conference Proceeding Volume I，pp. 257～260

[25]Wang Yan，Zhou Chunguang. Neural Network for Adaptive Routing Communacation Traffic in Wide Area Network. Beijing China：International Conference on neural Information Processing(ICONIP'95). oct. 1995，Volume 1of 2，pp. 585～588

[26]Yang Xiaowei，Liang Yanchun，Wang Zaishen，Zhou Chunguang. A Neural Network Algorithm of GCG Data Compression with Data Preprocessing. Beijing China：International Conference on Neural Information Processing(ICONIP'95). oct. 1995，Volume 1of 2，pp. 786～789

[27]Wang Yan，Zhou Chunguang，Xiao Yang，Zhou Guoqin. Solving The Adaptive Routing Problem in Wide Area Networks with The Improved Hopfied Neural Network. Beijing China：International Conference on Information Infrastructure ICII'96. Apr. 1996，Proceedings，pp. 113～116

第二篇

模糊系统



第六章　模糊数学基础

§6.1　概　　述

6.1.1　传统数学与模糊数学

长期以来，人们认为：一类知识，如果还不能用严格的数学方法来处理的话，那就不能被称之为科学．这种观念使人们养成了尊重精确、严格和定量的东西，而蔑视模糊和定性的东西的习惯．

传统数学是非常严格、非常精确的．传统数学的基础是集合论，要求集合的边界必须是明确的，即一个对象对于一个集合，要么属于，要么不属于，二者必居其一，绝不允许模棱两可．自然界和人类社会中的诸事万物都具有各种特征或属性，其中有些是严格而清晰的，但也有很多是模糊的．以人为例，人的性别（男或女）、年龄（岁数）、文化程度（文盲、小学、初中、高中、大专、大学、硕士、博士）等等明确特征都可以用传统数学的集合来描述，但像健康状况（好、比较好、良、……）、性格特点（聪明、能干、活泼、……）等这些模糊属性就很难用传统集合来描述．即使一些本来有严格定义的特征，为了便于处理，有时也更多地用模糊概念来描述，比如按年龄将人分为“年轻人”、“中年人”、“老年人”，传统数学方法是规定一些阈值来定义集合的界限，设用 y 代表年龄，$y<40$ 为“年轻人”，$40\leqslant y<60$ 为“中年人”，$y\geqslant 60$ 为“老年人”．这种方法简单，但过于绝对化，阈值界定是一个令人头痛的难题．正因为传统数学不能真实地描述和处理没有明确边界的模糊概念，模糊数学便应运而生．

模糊（Fuzzy），有“不分明”、“界限不清”之意，与清晰一词相对．概括地讲，模糊数学是用来描述、研究、处理事物所具有的模糊特征（即模糊概念）的数学．“模糊”是指它的研究对象，而“数学”是指它的研究方法，二者并不矛盾．

6.1.2　不相容原理

模糊数学诞生于 1965 年，美国自动化控制专家扎德（Zadeh L A）教授首先提出用隶属度函数（membership function）来描述模糊概念，创立了模糊集合论，为模糊数学奠定了基础．他还提出了著名的复杂性与精确性的“不相容原理”（又叫做“互斥性原理”），即“随着系统复杂性的增加，我们对其特性做出精确而有意义的描述的能力会随之降低，直到达到一个阈值，一旦超过它，精确和有意义二者将会相互排斥”．这就是说，事物越复杂，人们对它的认识也就越模糊，也就越需要模糊数学．不相容原理深刻地阐明了模糊数学产生和发展的必然性，也为三十多年来模糊数学的发展历史所证实．

§6.2　模糊集合与隶属度函数

6.2.1　模糊集合及其运算

一、模糊集合(Fuzzy Sets)的定义

传统集合论对事物只用"1"和"0"作简单的"属于"和"不属于"的分类，集合中的元素是有精确特性的对象，称之为普通集合. 例如，"8 到 12 之间的实数"是一个精确集合 C，$C=\{实数\ r|8\leqslant r\leqslant 12\}$，用特征函数 $\mu_C(r)$表示其成员，如图 6.1(a)所示.

$$\mu_C(r)=\begin{cases}1, & 8\leqslant r\leqslant 12\\ 0, & 其他\end{cases}$$

扎德把传统的普通集合扩充为模糊集合. 在模糊论域中，论域上的元素符合程度不是绝对的 0 或 1，而是介于 0 和 1 之间的一个实数. 集合中的元素的特性是模糊的. 例如，"接近 10 的实数"是一个模糊集合 $F=\{r|接近\ 10\ 的实数\}$，用"隶属度(Membership)"$\mu_F(r)$作为特征函数来描述元素属于集合的程度，如图 6.1(b)所示，10 属于 F 的隶属度为 1，9 和 11 属于 F 的隶属度为 0.75，7.2 和 12.8 属于 F 的隶属度为 0.275.

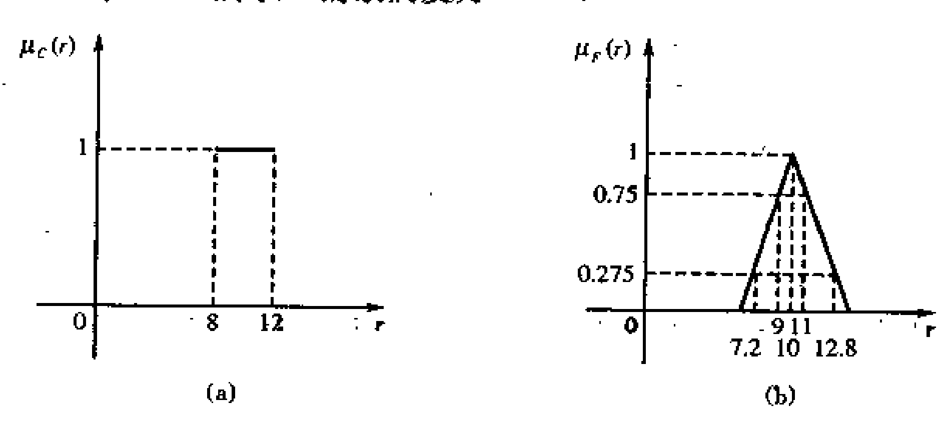


图 6.1　普通集合与模糊集合的对比

模糊集合的定义如下：论域 U 上的一个模糊集合 F 是指，对于论域 U 中的任一元素 $u\in U$，都指定了[0,1]闭区间中的一个数 $\mu_F(u)\in[0,1]$与之对应，$\mu_F(u)$称为 u 对模糊集合 F 的隶属度. 也可以表示成映射关系：

$$\mu_F:U\to[0,1]$$
$$u\to\mu_F(u)$$

这个映射称为模糊集合 F 的隶属度函数(membership function). 模糊集合有时也称为模糊子集.

$\mu_F(u)$用来说明 u 隶属于 F 的程度. $\mu_F(u)=1$ 表示 u 完全属于 F，$\mu_F(u)=0$ 表示 u 完全不属于 F，$0<\mu_F(u)<1$ 表示 u 部分属于 F. U 中的模糊集合 F 可以用元素 u 及其隶属度 $\mu_F(u)$来表示：

$$F=\{(u,\mu_F(u))|u\in U\}$$

仍以前而提到的"年轻"、"中年"、"老年"为例，这三个年龄特征分别用模糊集合 A、B、C 表示，它们的论域都是 $U=[0,100]$，论域中的元素都是年龄 u，我们可以规定模糊集合 A、B、C

的隶属度函数分别为$\mu_A(u)$、$\mu_B(u)$、$\mu_C(u)$，如图6.2所示.

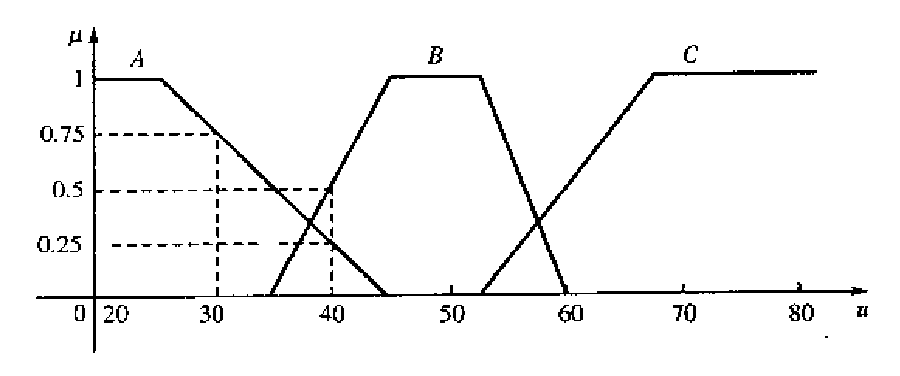


图6.2 “年轻”、“中年”、“老年”的隶属度函数

如果$u_1=30$，u_1对A的隶属度$\mu_A(u_1)=0.75$，这意味着30岁属于“年轻”的程度是0.75. 如果$u_2=40$，u_2既属于A又属于B，$\mu_A(u_2)=0.25$，$\mu_B(u_2)=0.5$，这说明40岁的人已不太年轻了，比较接近中年，但属于“中年”的程度还不太大，只有0.5. 再如$u_3=50$，$\mu_B(u_3)=1$，这说明50岁正值“中年”，但即将走向“老年”. 对比传统数学用阈值划分为三个年龄段的方法，显然用模糊集合能够比较准确地、真实地描述人们头脑中的原有概念，而用普通集合来描述模糊性概念反倒是不准确、不真实的，也可以说是粗糙的.

模糊集合F的支集S是一个普通集合，它由论域U中满足$\mu_F(u)>0$的所有u组成. 即$S=\{u\in U|\mu_F(u)>0\}$. 例如，在图6.2中，模糊集合B(“中年”)的支集是闭区间[35,60].

二、模糊集合的表示

1. 离散论域

如果论域U中只包含有限个元素，该论域称为离散论域. 设离散论域$U=\{u_1,u_2,\cdots,u_n\}$，U上的模糊集合F可表示为

$$F=\sum_{i=1}^{n}\mu_F(u_i)/u_i=\mu_F(u_1)/u_1+\mu_F(u_2)/u_2+\cdots+\mu_F(u_n)/u_n \tag{6.2.1}$$

这只是一种表示法，表明对每个元素u_i所定义的隶属度为$\mu_F(u_i)$，并不是通常的求和运算.

2. 连续论域

如果论域U是实数域，即$U\in\mathbf{R}$，论域中有无穷多个连续的点，该论域称为连续论域. 连续论域上的模糊集合可表示为

$$F=\int_{U\in\mathbf{R}}\mu_F(u)/u \tag{6.2.2}$$

这里的积分号也不是通常的含义，该式只是表示对论域中的每个元素u都定义了相应的隶属度函数$\mu_F(u)$.

三、模糊集合的基本运算

1. 基本运算的定义

设A，B是同一论域U上的两个模糊集合，它们之间包含、相等关系定义如下：

· A包含B，记做$A\supseteq B$，有

$$\mu_A(u)\geqslant\mu_B(u),\quad\forall u\in U \tag{6.2.3}$$

· A等于B，记做$A=B$，有

$$\mu_A(u)=\mu_B(u),\quad \forall u\in U \tag{6.2.4}$$

显然,$A=B\Leftrightarrow A\supset B$ 且 $A\subset B$.

设 A、B 是同一论域 U 上的两个模糊集合,隶属度函数分别为 $\mu_A(u)$ 和 $\mu_B(u)$,它们的并、交、补运算定义如下:

· A 与 B 的交,记做 $A\cap B$,有

$$\mu_{A\cap B}(u)=\mu_A(u)\wedge\mu_B(u)=\min\{\mu_A(u),\mu_B(u)\},\quad \forall u\in U \tag{6.2.5}$$

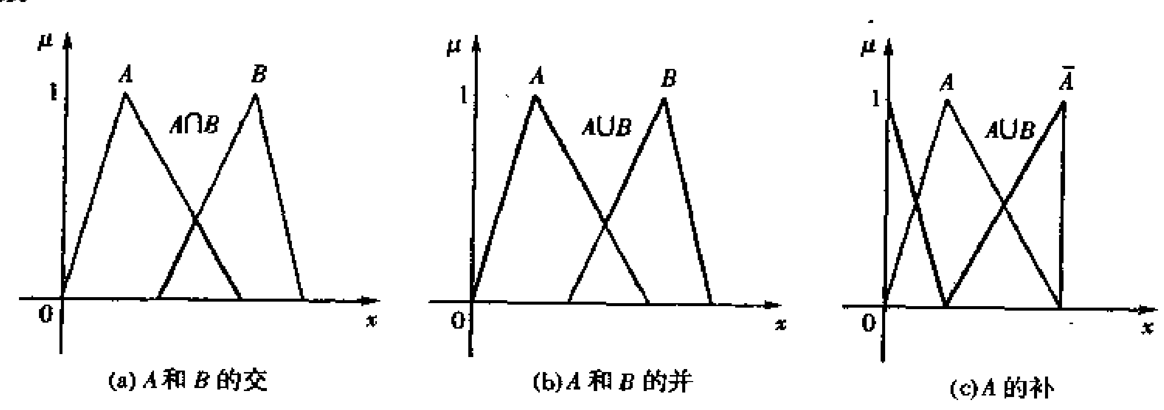
· A 与 B 的并,记做 $A\cup B$,有

$$\mu_{A\cup B}(u)=\mu_A(u)\vee\mu_B(u)=\max\{\mu_A(u),\mu_B(u)\},\quad \forall u\in U \tag{6.2.6}$$

· A 的补,记做 $\overline{A}$,有

$$\mu_{\overline{A}}(u)=1-\mu_A(u),\quad \forall A\in U \tag{6.2.7}$$

其中,min 和 $\wedge$ 表示取小运算,max 和 $\vee$ 表示取大运算.图 6.3 显示了这三种运算对应的隶属度函数.



(a) A 和 B 的交　(b) A 和 B 的并　(c) A 的补

图 6.3 模糊集合的三种运算

例题:设论域 $X=\{x_1,x_2,x_3,x_4\}$ 上的模糊集合 A,B 分别是

$$A=\frac{1}{x_1}+\frac{0.9}{x_2}+\frac{0.4}{x_3}+\frac{0.2}{x_4}$$

$$B=\frac{0.5}{x_1}+\frac{0.8}{x_2}+\frac{0}{x_3}+\frac{0.3}{x_4}$$

求 $A\cap B,A\cup B,\overline{A},\overline{B}$.

解

$$A\cup B=\frac{1\vee 0.5}{x_1}+\frac{0.9\vee 0.8}{x_2}+\frac{0.4\vee 0}{x_3}+\frac{0.2\vee 0.3}{x_4}=\frac{1}{x_1}+\frac{0.9}{x_2}+\frac{0.4}{x_3}+\frac{0.3}{x_4}$$

$$A\cap B=\frac{1\wedge 0.5}{x_1}+\frac{0.9\wedge 0.8}{x_2}+\frac{0.4\wedge 0}{x_3}+\frac{0.2\wedge 0.3}{x_4}=\frac{0.5}{x_1}+\frac{0.8}{x_2}+\frac{0}{x_3}+\frac{0.2}{x_4}$$

$$\overline{A}=\frac{1-1}{x_1}+\frac{1-0.9}{x_2}+\frac{1-0.4}{x_3}+\frac{1-0.2}{x_4}=\frac{0}{x_1}+\frac{0.1}{x_2}+\frac{0.6}{x_3}+\frac{0.8}{x_4}$$

$$\overline{B}=\frac{1-0.5}{x_1}+\frac{1-0.8}{x_2}+\frac{1-0}{x_3}+\frac{1-0.3}{x_4}=\frac{0.5}{x_1}+\frac{0.2}{x_2}+\frac{1}{x_3}+\frac{0.7}{x_4}$$

2. 基本运算定律

论域 U 上的模糊全集 E 和模糊空集 φ 定义如下：

$$\mu_E(u)=1,\quad \forall u\in U \tag{6.2.8}$$

$$\mu_\phi(u)=0,\quad \forall u\in U \tag{6.2.9}$$

设 A,B,C 是论域 U 上的三个模糊集合，它们的交、并、补运算有下列定律：

①恒等律：

$$A\cup A=A, A\cap A=A$$

②交换律：

$$A\cup B=B\cup A, A\cap B=B\cap A$$

③结合律：

$$(A\cup B)\cup C=A\cup(B\cup C)$$

$$(A\cap B)\cap C=A\cap(B\cup C)$$

④分配律：

$$A\cup(B\cap C)=(A\cup B)\cap(A\cup C)$$

$$A\cap(B\cup C)=(A\cap B)\cup(A\cap C)$$

⑤吸收律：

$$(A\cap B)\cup A=A$$

$$(A\cup B)\cap A=A$$

⑥同一律：

$$A\cup E=E, A\cap E=A$$

$$A\cup\phi=A, A\cap\phi=\phi$$

⑦复原律：

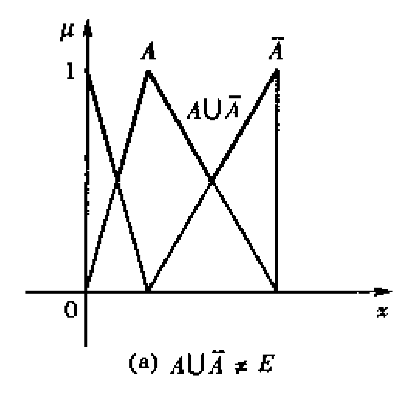
$$\overline{\overline{A}}=A$$

⑧对偶律(摩根律)：

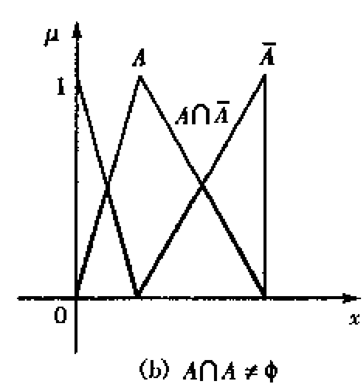
$$\overline{A\cup B}=\overline{A}\cap\overline{B}$$

$$\overline{A\cap B}=\overline{A}\cup\overline{B}$$

以上八条运算定律，模糊集合和普通集合是完全相同的，但是普通集合的"互补律"对模糊集合却不成立，如图 6.4 所示，即



(a) $A\cup\overline{A}\neq E$



(b) $A\cap A\neq\phi$

图 6.4　模糊集合的运算不满足"互补律"

$$A\cup\overline{A}\neq E,\quad A\cap\overline{A}\neq\phi$$

四、模糊关系

设有两个集合 A,B，A 和 B 的直积 $A\times B$ 定义为

$$A\times B=\{a,b\mid a\in A,b\in B\}$$

它是由序偶 (a,b) 的全体所构成的二维论域上的集合. 一般来说 $A\times B\neq B\times A$.

设 $A\times B$ 是集合 A 和 B 的直积，以 $A\times B$ 为论域的模糊集合 R 称为 A 和 B 的模糊关系. 也就是说对 $A\times B$ 中的任一元素 (a,b)，都指定了它对 R 的隶属度 $\mu_R(a,b)$，R 的隶属度函数 μ_R 可看做是如下的映射：

$$\mu_R: A\times B\rightarrow[0,1]$$
$$(a,b)\rightarrow\mu_R(a,b)$$

设 R_1 是 X 和 Y 的模糊关系，R_2 是 Y 和 Z 的模糊关系，那么 R_1 和 R_2 的合成是 X 到 Z 的一个模糊关系，记做 $R_1\circ R_2$，其隶属度函数为

$$\mu_{R_1\circ R_2}(x,z)=\bigvee_{y\in Y}\left[\mu_{R_1}(x,y)\wedge\mu_{R_2}(y,z)\right],\quad\forall(x,z)\in X\times Z\tag{6.2.10}$$

6.2.2 隶属度函数

模糊集合完全由其隶属度函数所刻画. 若隶属度函数的取值只取 0 和 1，那么模糊集合就蜕化为普通集合. 所以普通集合是模糊集合的特例，模糊集合是普通集合的推广.

正确地确定隶属度函数，是运用模糊集合解决实际问题的基础，是能否用好模糊集合的关键. 然而目前确定隶属度函数还没有一种成熟有效的方法，仍然停留在依靠经验确定，然后再通过实验进行修正. 这种方法经过人脑的加工，吸收了人脑的优点，但这与人确定的心理过程有关，带有一定的盲目性和主观性. 目前隶属度函数的确定方法大致有以下几种：

①模糊统计方法：用对样本统计实验的方法确定隶属度函数.

②例证法：从有限个元素的隶属度值来估计模糊子集隶属度函数.

③专家经验法：根据专家的经验来确定隶属度函数.

④机器学习法：通过神经网络的学习训练得到隶属度函数.

这些方法不是从根本上解决问题的一般性方法. 但从不同角度提出的确定方法，在解决和处理实际的模糊信息问题时却能殊途同归. 因此说隶属度函数的确定并不是惟一的，允许有不同的组合. 至于是否有、或者如何确定和证明最优的隶属度函数，还需要模糊理论家的进一步研究和探索. 目前常用的隶属度函数有：

1. 三角形

三角形隶属度函数曲线如图 6.5 所示，隶属度函数的解析式为

$$\mu_F(x)=\begin{cases}\dfrac{x-b}{a-b}, & b\leqslant x\leqslant a\\ \dfrac{c-x}{c-a}, & a<x\leqslant c\\ 0, & x<b\text{ 或 }x>c\end{cases}\tag{6.2.11}$$

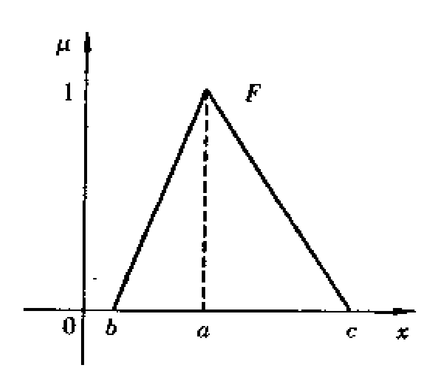


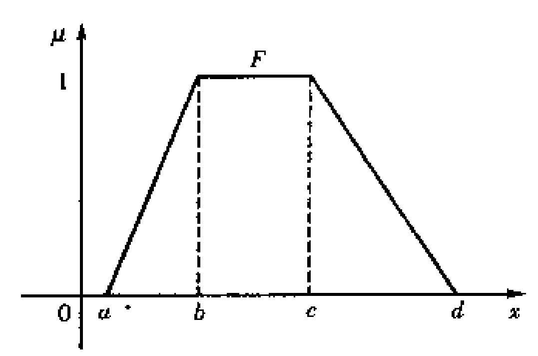
图 6.5　三角形隶属度函数图

2. 梯形

梯形与三角形是最简单的两种隶属度函数，应用也非

常广泛.梯形隶属度函数如图 6.6 所示,解析式表示为

$$\mu_F(x)=\begin{cases}\dfrac{x-a}{b-a}, & a\leqslant x\leqslant b\\ 1, & b\leqslant x\leqslant c\\ \dfrac{d-x}{d-c}, & c< x\leqslant d\\ 0, & x<a \text{ 或 } x>d\end{cases} \tag{6.2.12}$$



6.6 梯形隶属度函数

3. 正态型

这是一种最主要、最常见的分布,表示为:

$$\mu(x)=\exp\left\{-\left(\frac{x-a}{b}\right)^2\right\},\quad b>0 \tag{6.2.13}$$

其分布曲线如图 6.7 所示:

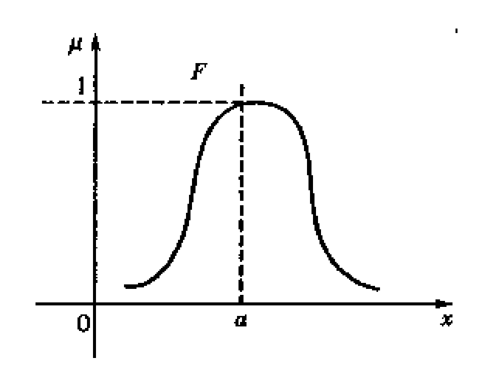


图 6.7 正态型分布曲线

4. Γ 型

如图 6.8 所示,解析式表示为:

$$\mu_F(x)=\begin{cases}0, & x<0\\ \left(\dfrac{x}{\lambda\nu}\right)^{\nu}\cdot e^{\nu-\frac{x}{\lambda}}, & x\geqslant 0\end{cases} \tag{6.2.14}$$

其中 $\lambda>0,\nu>0$.

5. Sigmiod 型

如图 6.9 所示,解析式为:

$$\mu_F(x)=\frac{1}{1+e^{-x}} \tag{6.2.15}$$

最后,需要注意隶属度与概率之间关系,虽然二者研究的都是事物的不确定性问题,在

[0,1]区间取值,但概率研究的事件本身是清晰的,只是事件出现的频率不确定;而隶属度正好相反,它研究的事件本身是模糊的,事件出现的频率是确定的.

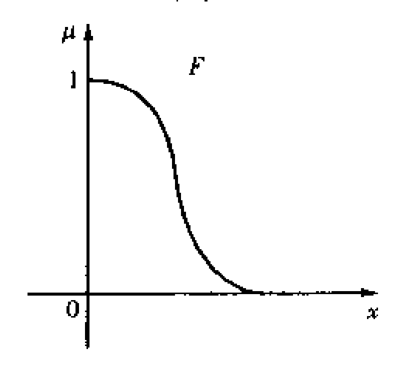
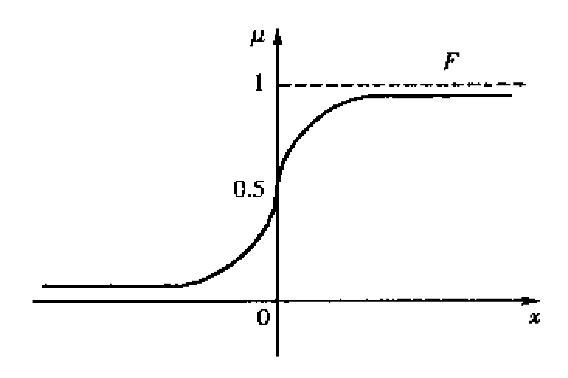


图 6.8　Γ型隶属度函数图



6.9　Sigmoid 型隶属度函数

§6.3　模糊逻辑与模糊推理

6.3.1　模糊逻辑

逻辑是探索、阐述和确立有效推理原则的一门学科,是研究人的思维形式和思维规律的.在历史上逻辑学与数学相互渗透形成了一门新的学科——数理逻辑.数理逻辑采用符号进行逻辑推理,它与经典集合论相对应,在逻辑上只取真假二值,因此,成为计算机科学的基础理论之一.人们通常称普通逻辑为二值逻辑,用来表达一个完整概念的语言或文字符号称为句子.对于一个句子,可以判断其真假时,称该句子为命题.一个命题的真或假称为该命题的真值,通常用 0 或 1 表示.当命题为真时,取真值为 1;当命题为假时,取真值为 0.

在现实当中一些命题含有模糊谓词,这样的命题称为模糊命题.模糊命题不能用绝对的真假来刻画,只能讨论其真假程度如何.因此,不能只用 0、1 二值来描述.很自然想到模糊命题的真值应是隶属度函数,其取值应扩大到区间[0,1]上连续取值.模糊命题是普通命题在概念上的拓广.它对应的逻辑是连续逻辑(或多值逻辑),又称为模糊逻辑.显然,不仅普通命题能反映客观世界,而模糊命题更是现实生活中常见的.随着模糊逻辑的出现和发展,将对计算机科学、人工智能、模糊控制等方向的研究和发展起推动作用.

下面对模糊逻辑的运算作一简单介绍.

设有模糊命题 X 和 Y,对应的真值(隶属度,也称为模糊变量)$x,y\in[0,1]$,称:

①$X\wedge Y$ 为模糊逻辑合取(交、与),真值为 $x\wedge y=\min(x,y)$.

②$X\vee T$ 为模糊逻辑析取(并、或),真值为 $x\vee y=\max(x,y)$.

③$\bar{X}$ 为模糊逻辑否定(补、非),真值为 $\bar{x}=1-x$.

④$X\rightarrow Y$ 为模糊逻辑蕴含,真值为 $x\rightarrow y=\bar{x}\vee y$.

⑤$X\leftrightarrow Y$ 为模糊逻辑恒等,真值为 $x\leftrightarrow y=(\bar{x}\vee y)\wedge(\bar{y}\vee x)$.

5.3.2　语言变量

模糊逻辑原则上是一种模拟人的思维的逻辑.要用从 0 到 1 的区间上的确切数值来表示一个模糊命题的真假程度,有时是很困难的.人们在日常生活中相互交流时,使用的是自然语

言. 自然语言是以字为符号,以词句为单位的,用来表达主客观世界的各种事物、观念、行为和情感的意义,是人们思维和交流信息的重要工具,其特点是不确定性,如“今天是个好天气”、“机器太旧了”等. 这种带有模糊性的语言,也称为模糊语言. 要用计算机来模仿人的思维、推理和判断,表达模糊语言,就必须引进语言变量. 扎德教授在 1975 年提出了语言变量的概念,语言变量实际上是一种模糊变量,它用词句而不是用数字来表示变量的“值”. 引进语言变量后,就构成了模糊语言逻辑.

一、模糊数与语言变量

模糊集合的应用为系统地处理不清晰、不精确概念的方法提供了基础,可以应用模糊集合来表示语言变量. 然而语言变量既可以用模糊数来表示,也可以用语言形式术语来定义. 模糊数和语言变量的定义如下:

连续论域 U 中的模糊数 F 是一个 U 上的正规凸模糊集合. 这里所谓正规集合的含义就是其隶属度函数的最大值是 1,即 $\max\limits_{u\in U}\mu_F(u)=1$. 这里的凸集合的含义是:在隶属度函数曲线上任意两点之间,曲线上的任意一点所表示的隶属度都大于或者等于两点隶属度中较小的一个,即在实数集合的任意区间$[a,b]$上,对于所有的 $x\in[a,b]$,都有:

$$\mu_F(x)\geqslant\min(\mu_F(a),\mu_F(b)),a,b\in U,x\in[a,b] \tag{6.3.1}$$

语言变量用一个有五个元素的集合$(N,T(N),U,G,M)$来表征,其中

(1)N 是语言变量的名称,如年龄、数的大小等;

(2)U 为语言变量 N 的论域;

(3)$T(N)$为语言变量的值 X 的集合,其中每个 X 都是论域 U 上的模糊集合,如:

$T(N)=T$(年龄)=“很年轻”+“年轻”+“中年”+“较老”+“很老”=

$$X_1+X_2+X_3+X_4+X_5$$

(4)G 为语法规则,用于产生语言变量 N 的值 X 的名称,研究单词构成、合成词后词义的变化,并求取其隶属度函数. 其中,用“或”、“与”、“非”作连接词构成的合成词,可以按模糊逻辑运算取真值,见 6. 3. 1 节;带修饰词算子的合成词,其真值可以根据经验公式计算出来. 常用的算子有以下几种:

①语气算子,如“很”、“略”、“相当”等;

②模糊化算子,如“大概”、“近乎”、“差不多”等;

③判定化算子,如“偏向”、“多半是”、“倾向于”等.

(5)M 是语义规则,根据语义规则给出模糊子集 X 的隶属度函数.

以 N=年龄为例,表征语言变量的五元素集合如图 6.10 所示.

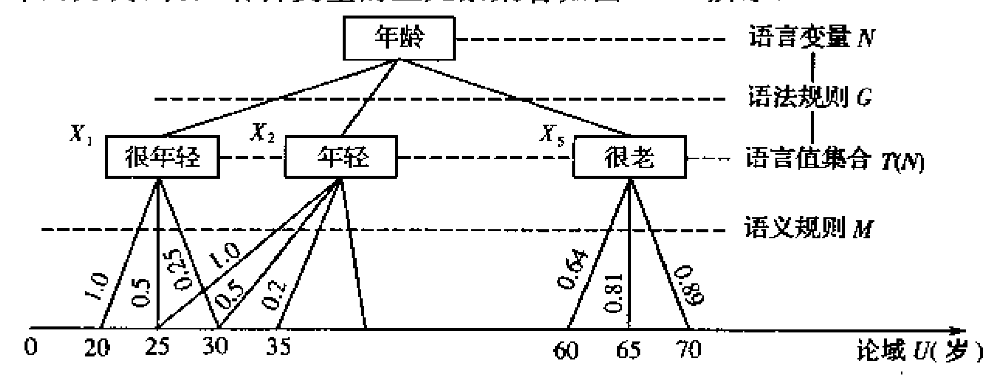


图 6.10 表示年龄的语言变量

例　Zadeh L A 在论域 $U=[0,100$ 岁$]$内给出了年龄的语言变量值"老"的模糊子集隶属度函数为

$$\mu_{老}(x)=\begin{cases}0, & x<50\\ \dfrac{1}{1+\left(\dfrac{x-50}{5}\right)^{-2}}, & x\geqslant 50\end{cases}$$

其中修饰词的隶属度函数为：$\mu_{极A}=A_A^4$，$\mu_{非常A}=\mu_A^2$，$\mu_{相当A}=\mu_A^{1.25}$，$\mu_{比较A}=\mu_A^{0.75}$，$\mu_{略A}=\mu_A^{0.5}$，$\mu_{稍微A}=\mu_A^{0.25}$.

现以 60 岁为例，通过隶属度函数分别计算它属于"极老"、"非常老"、"相当老"、"比较老"、"略老"、"稍微老"的程度为

$$\mu_{极老}(60)=[\mu_{老}(60)]^4=(0.8)^4=0.41$$
$$\mu_{非常老}(60)=[\mu_{老}(60)]^2=(0.8)^2=0.64$$
$$\mu_{相当老}(60)=[\mu_{老}(60)]^{1.25}=(0.8)^{1.25}=0.757$$
$$\mu_{比较老}(60)=[\mu_{老}(60)]^{0.75}=(0.8)^{0.75}=0.845$$
$$\mu_{略老}(60)=[\mu_{老}(60)]^{0.5}=(0.8)^{0.5}=0.89$$
$$\mu_{稍微老}(60)=[\mu_{老}(60)]^{0.25}=(0.8)^{0.25}=0.946$$

二、模糊语句

1. 模糊直言语句

模糊直言语句的句型为"x 是 A"，其中 x 是对象的名称，A 是论域 U 上的一个模糊子集.

2. 模糊条件语句

常用的模糊条件语句的句型有：

①"若 A 则 B"型，也记为 if A then B；

②"若 A 则 B 否则 C"型，也记为 if A then B else C；

③"若 A 且 B 则 C"型，也记为 if A and B then C.

6.3.3　模糊推理

模糊推理是不确定性推理方法的一种，它是一种以模糊判断为前提，运用模糊语言规则，推出一个近似的模糊判断结论的方法，其基础是模糊逻辑，它是在二值逻辑三段论的基础上发展起来的. 虽然模糊推理的理论不是很成熟，但是在应用实践中证明是有效的，并且用这种推理方法得到的结论与人的思维一致或接近.

模糊推理的两种重要推理规则：

(1)广义前向推理法(Generalize Modus Ponens，简称 GMP)

　前提 1：如果 x 是 A，则 y 是 B

　前提 2：x 是 A'

　结论：那么 y 是 B'

(2)广义后向推理法(Generalize Modus Tollens，简称 GMT)

　前提 1：如果 x 是 A，则 y 是 B

　前提 2：y 是 B'

　结论：那么 x 是 A'

1975 年 Zadeh 利用模糊变换关系，在广义前向推理法的基础上，提出了模糊逻辑推理的合成规则，建立了统一的数学模型，用于对各种模糊推理作统一处理. 其推理规则为：

前提：如果 x 是 A，则 y 是 B

事实：x 是 A'

结论：那么 y 是 $B'=A'\circ(A\rightarrow B)$

即结论 B' 可用 A' 与由 A 到 B 的推理关系进行合成而得到，其中的算子"$\circ$"表示模糊关系的合成运算，$(A\rightarrow B)$ 表示由 A 到 B 进行推理的关系或者条件，即"如果 x 是 A，那么 y 是 B"的简化表示方法. 有时 $(A\rightarrow B)$ 也可写成 $R_{A\rightarrow B}$，其隶属度函数被定义为

$$\mu_{A\rightarrow B}(x,y)=\mu_A(x)\rightarrow\mu_B(y) \tag{6.3.2}$$

那么 $B'=A'\circ(A\rightarrow B)$ 的隶属度函数为

$$\mu_{B'}(y)=\bigvee_x\{\mu_{A'}(x)\wedge\mu_{A\rightarrow B}(x,y)\}=\bigvee_x\{\mu_{A'}(x)\wedge[\mu_A(x)\rightarrow\mu_B(y)]\} \tag{6.3.3}$$

如何实现合成运算，有各种不同的方法，这决定于对蕴含运算的定义.

一、Zadeh 模糊假言推理法

Zadeh 把 $(A\rightarrow B)$ 定义成

$$(A\rightarrow B)=1\wedge(1-A+B)\text{或者}(A\rightarrow B)=(A\wedge B)\vee(1-A) \tag{6.3.4}$$

对于后者，其隶属度函数为

$$\mu_{A\rightarrow B}(x,y)=\mu_A(x)\rightarrow\mu_B(y)=[\mu_A(x)\wedge\mu_B(y)]\vee[1-\mu_A(x)] \tag{6.3.5}$$

把(6.3.5)代入(6.3.3)式，便得到了结论 B' 的隶属度函数.

二、Mamdani 推理法

Mamdani 则把 $(A\rightarrow B)$ 定义成 $(A\rightarrow B)=A\wedge B$. 下面是 Mamdani 推理法的具体过程.

设 $U_1,U_2,\cdots,U_n$ 为 n 个有界论域，记 $U_i=[a_i,b_i]$. 每个论域按一定规则分为 l_i 个凸模糊子集 A_{ij}，其隶属度函数记为 $\mu_{A_{ij}}(x_i)$. 记 $S_i=\{A_{ij}|,j=1,2,\cdots,l_i\}$. 则我们将模糊规则集表示为：

$$\mathop{\mathrm{OR}}_{j=1}^{m}(\mathrm{IF}(\mathop{\mathrm{AND}}_{i=1}^{n}(x_i\mathrm{ISA}_{ij}))\mathrm{THEN}\ Y\ \mathrm{IS}\ B_j \tag{6.3.6}$$

其中 m 为模糊规则数，n 为输入变量个数，$A,B\in S_i$.

如果有事实"if x_1 is a_1 and x_2 is a_2 and…x_n is a_n"，则结论"Y is B'"可以这样得出：由前提和第 j 条模糊规则可得到推理结果为 B'_j，则

$$M_{B'_j}(z)=\mu_{A_{1j}}(a_1)\wedge\mu_{A_{2j}}(a_2)\wedge\cdots\wedge\mu_{A_{nj}}(a_n)\wedge\mu_{B_j}(z) \tag{6.3.7}$$

其中 $j=1,2\cdots m$，"$\wedge$"表示 min 操作.

经(6.3.6)式推理后的结论 B' 可综合推理结果 $B'_1,B'_2,\cdots B'_m$，得到：

$$\mu_{B'}(z)=\mu_{B'_1}(z)\vee\mu_{B'_2}(z)\vee\cdots\vee\mu_{B'_m}(z) \tag{6.3.8}$$

其中"$\vee$"表示 max 操作.

最终系统的输出可以由"重心法"求出：

$$z_0=\frac{\sum_{i=1}^{m}\mu_B(z_i)z_i}{\sum_{i=1}^{m}\mu_B(z_i)},\quad z_i\text{ 为常数} \tag{6.3.9}$$

图 6.11 所示的是规则数为 3($m=3$)，变量个数为 2($n=2$)的 Mamdani 推理过程.

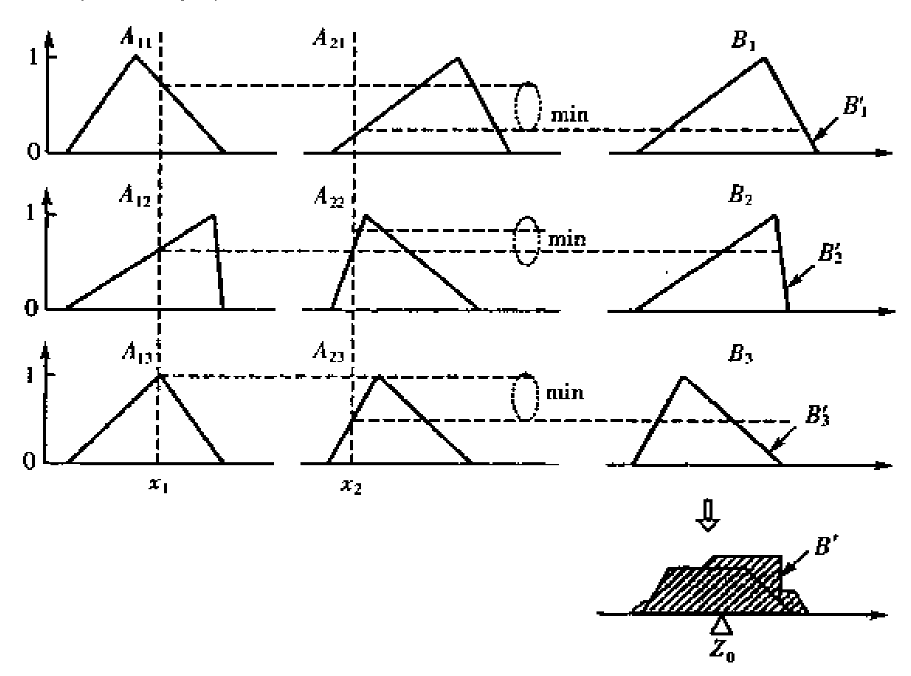


图 6.11　Mamdani 推理过程

三、模糊加权推理法

在模糊加权型推理法中，模糊规则集的结论表示为 w_j/z_j，即将式(6.3.6)表示为：

$$\mathop{\mathrm{OR}}_{j=1}^{m}(\mathrm{IF}(\mathop{\mathrm{AND}}_{i=1}^{n}(x_1 \text{ IS } A_{ij}))\mathrm{THEN}\ Y \text{ IS } w_j/z_j) \tag{6.3.10}$$

其中 m 为模糊规则数，n 为输入变量个数，$A\in S_i$，z_j 为常数. w_j/z_j 是构成模糊集合的一个元素，w_j 表示权重，并不表示 z_j 的等级，应看做模糊规则自身的加权，即模糊规则的重视度或重要度. 当 $w_j>1$ 时，表示对第 j 条规则的强调，当 $0\leqslant w_j<1$ 时，表示对第 j 条规则的抑制.

将推理结果中的 ∧ 运算改为"·"运算，我们定义事实"x_1 is a_1 and x_2 is a_2 and…x_n is a_n"和各模糊规则的前件的适合度为：

$$\mu_j(x)=\mu_{A_{1j}}(a_1)\cdot\mu_{A_{2j}}(a_2)\cdot\cdots\cdot\mu_{A_{nj}}(a_n),\quad j=1,2,\cdots,m \tag{6.3.11}$$

则最终的结论 z_0 可将规则后件 z_j 在各适合度 μ_j 中带上权重 w_j，由加权平均法求得，即：

$$z_0=\frac{\sum_{i=1}^{m}\mu_i w_i z_i}{\sum_{i=1}^{m}\mu_i w_i} \tag{6.3.12}$$

四、广义模糊加权推理法

在实际应用领域，模糊规则一般无法直接得到，我们希望建立一个自适应的模糊系统，实现模糊规则的自动提取和隶属度函数的自动改进. 为此我们改进了模糊加权型推理法的推理形式，提出了广义模糊加权型推理法. 在第八章将介绍基于此推理法的模糊神经网络，及其在味觉信号识别中的应用.

定义输入变量 x_i 的模糊子集数为 k_i，输出变量 Y 的模糊子集数为 l，设 $m=\prod_{i=1}^{n}k_i$，则模糊规则的最大条数为 $m\cdot l$. 试列举出模糊规则的各种可能形式，并用各规则对不同结果的权重

来表示推理结果的各种可能性. 即将式(6.3.10)中规则的结论变为 $w_{j1}/z_1, w_{j2}/z_2, \cdots, w_{jl}/z_l$，则模糊规则集可表示为

$$\mathop{\mathrm{OR}}_{j=1}^{m}(\mathrm{IF}(\mathop{\mathrm{AND}}_{i=1}^{n}(x_i \text{ IS } A_{ij}))\mathrm{THEN}\ Y \text{ IS } w_{j1}/z_1, w_{j2}/z_2, \cdots, w_{jl}/z_l) \tag{6.3.13}$$

其中 m 为模糊规则数，n 为输入变量个数，l 为输出变量模糊子集个数，z_i 为常数，是结论隶属度函数的支集值. w_{ji} 表示权重. 定义事实和各模糊规则前件的适合度为 μ_j：

$$\mu_j(z) = \mu_{A_{1j}}(a_1) \cdot \mu_{A_{2j}}(a_2) \cdot \cdots \cdot \mu_{A_{nj}}(a_n), \quad j=1,2\cdots,m \tag{6.3.14}$$

则最终的结论 z_0 可由下面改进的加权平均法求得：

$$z_0 = \frac{\sum_{i=1}^{l}\left(f\left(\sum_{j=1}^{m}\mu_j w_{ji}\right)\cdot z_i\right)}{\sum_{i=1}^{l}\left(f\left(\sum_{j=1}^{m}\mu_j w_{ji}\right)\right)} \tag{6.3.15}$$

其中 $f(x)$ 可取 Sigmoid 形函数，

$$f(x) = \frac{1}{1+\exp(-\alpha\cdot(x-\beta))}, \quad \alpha、\beta \text{ 为常数} \tag{6.3.16}$$

第七章　模糊控制理论

§7.1　模糊控制原理

7.1.1　模糊控制

经典控制论在军事、工业、航天等领域都取得了很大成就.在应用控制理论的时候,必须首先对研究对象建立精确的数学模型.但是,在许多复杂的控制系统中,例如复杂的工业系统、大规模的交通控制系统、模式识别系统等由于系统的复杂性,要建立精确的数学模型是困难的.至于像生物学、经济学、心理学等领域中,传统的数学方法更难进行.由于这些系统呈现出各种不确定性,使得模糊数学成为研究这类系统的一个有力工具.

正是因为如此,模糊控制论作为模糊数学的一个分支,就应运而生.1974年,英国工程师Mamdani首先把模糊集合用于锅炉蒸汽机的控制上,并发表了模糊控制论方面的第一篇论文.这就标志着模糊控制的诞生.此后,许多国家都开展了这方面的工作,并在理论和应用上都取得了可喜的成果.

模糊控制是通过模糊控制器来实现的.模糊控制器是对人脑所具有的模糊推理机能的模拟.应用模糊数学的知识,模拟人的思维方法,把人用自然语言描述的控制策略改造成模糊控制规则,把输入输出作为模糊集合按模糊描理的方法进行处理,进面确定控制量.

先来了解一下人脑是如何进行模糊控制的,可以通过考察一个有经验的操作人员的控制过程,如航海驾驶员,当他发现航船已偏离航线时,实际上偏离多少是准确值,但反映到驾驶员头脑中(输入)只能是模糊概念,凭经验进行判断和决策(经验就是模糊控制规则),决定操作,即进行打舵(输出).输出虽然是模糊的,但操作却是准确的.

根据上而的形象分析,实现模糊控制的过程可分成下面三步来完成。

一、模糊化

当用模糊控制器来代替人进行模糊控制时,由于控制器本身没有思维,需要把观测值(通常是精确数字)转化为相应的模糊集合,这个转化就是模糊化,让模糊控制器去处理.为了实现模糊化先要进行论域变换,将真实论域变换为内部论域,并要针对输入语言变量定义模糊于集及其隶属度函数.

二、模糊控制规则的建立和模糊推理

通过一组模糊条件语句构成模糊控制规则,并计算模糊控制规则决定的模糊关系,然后根据推理合成规则进行模糊推理.在这一步,输人语言变量被加到一个 if…then…控制规则的集合中去,把各种规则的结果加在一起产生一个"模糊输出"集合.

模糊控制规则通常是将操作者在控制过程中的实践经验(即手动控制策略)加以总结面得到的一条条模糊条件语句(形如"if…then…"),它是模糊控制的核心.

模糊控制规则集是由一组被此间通过"或"关系连接起来的模糊条件语句来描述的.其中每一条模糊条件语句,当输入、输出语言变量在各自论域上对各模糊子集的隶属度为已知时,

都可以表达为论域直积上的模糊关系. 在计算出每一条模糊语句决定的模糊关系 $R_i(i=1,2,\cdots,m)$(其中 m 为一组模糊条件语句中的语句数)之后,考虑到这些模糊条件语句间是"或"的关系,可得描述整个系统的控制规则的总模糊关系 R 为

$$R = R_1 \cup R_2 \cup \cdots \cup R_m = \bigcup_{i=1}^{m} R_i \tag{7.1.1}$$

三、解模糊判决

通过模糊推理得到的结果是一个模糊集合或者隶属度函数,但在实际应用中,往往需要的是一个确定的值. 在推理得到的模糊集合中取一个最能代表这个模糊集合的单值的过程叫做解模糊判决或者非模糊化(Defuzzication). 解模糊判决可以采用不同的方法,不同方法所得到的结果是不同的. 理论上用重心法比较合理,但是计算比较复杂,故在实时性要求高的系统中不采用这种方法. 最简单的方法是取大隶属度方法,但这种方法未考虑其他隶属度较小的那些值的影响,代表性不好,所以经常用于简单的系统. 介于这两者之间的还有各种平均法,如加权平均法、隶属度限幅(α-cut)元素平均法等.

1. 重心法

所谓重心法就是取模糊隶属度函数曲线与横坐标轴围成面积的重心作为代表点. 理论上说,应该计算输出范围内一系列连续点的重心,即

$$u = \frac{\int x\mu_N(x)\mathrm{d}x}{\int \mu_N(x)\mathrm{d}x} \tag{7.1.2}$$

但是在实际应用中,可以计算输出范围内若干离散值的重心,用足够小的采样间隔来提供所需要的精度,这是一种较好的折衷方案,即

$$u = \frac{\sum x_i \cdot \mu_N(x_i)}{\sum \mu_N(x_i)} \tag{7.1.3}$$

2. 最大隶属度法

这种方法最简单,只要在推理结论的模糊集合中取隶属度最大的那个元素作为输出量即可. 不过这种情况下其隶属度曲线一定是正规凸模糊集合(即曲线只能是单峰曲线). 另外,如果该曲线是梯形平顶的,那么具有最大隶属度的元素就可能不止一个,这时就要对所有取最大隶属度的元素求平均值.

3. 系数加权平均法

系数加权平均法的输出由下式决定,

$$u = \frac{\sum k_i x_i}{\sum k_i} \tag{7.1.4}$$

这里的系数 k_i 的选择要根据实际情况决定,不同的系数决定系统具有不同的响应特性. 当系数选择 $k_i=\mu_N(x_i)$ 时,即取其隶属度函数时,这就是重心法. 在模糊逻辑控制中,可以通过选择和调整该系数来改善系统的响应特性,这种方法具有较大的灵活性.

4. 隶属度限幅元素平均法

用所确定的隶属度值 α 对隶属度函数曲线进行切割,再对切割后等于该隶属度的所有元素进行平均,用这个平均值作为输出,这种方法称为隶属度限幅(α-cut)元素平均法.

7.1.2 模糊控制器的基本结构与工作原理

最简单的模糊控制器(Fuzzy Controler,简称 FC)的结构如图 7.1 所示. 虚线框内的就是模糊控制器(FC),要完成一次控制动作,首先将观测值输入模糊控制器,观测值在控制器内经过模糊化,根据模糊规则集进行模糊推理,得到的模糊响应经过解模糊判决后,输出确切响应,然后作用到被控对象上.

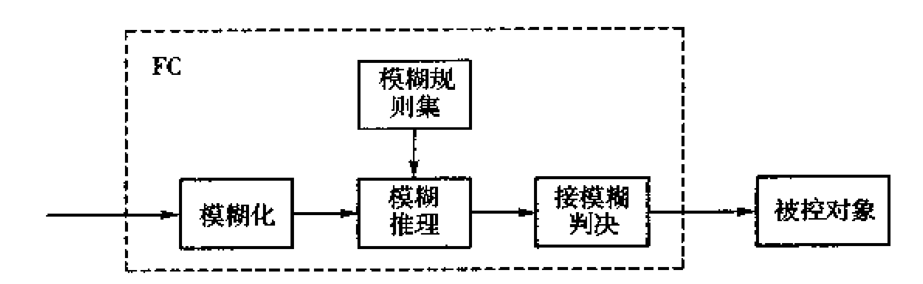


图 7.1 模糊控制器的示意图

由于模糊控制器的控制规则是根据操作者的手动控制经验总结出来的,而操作者一般只能观察到被控对象的输出变量及其变化率,故在模糊控制器中通常将误差及其变化作为输入语言变量,而将控制量的变化作为输出语言变量,并作为被控对象的输入变量. 以系统误差及其变化率作为输入语言变量,以控制量的变化作为输出语言变量的模糊控制器结构如图 7.2 所示.

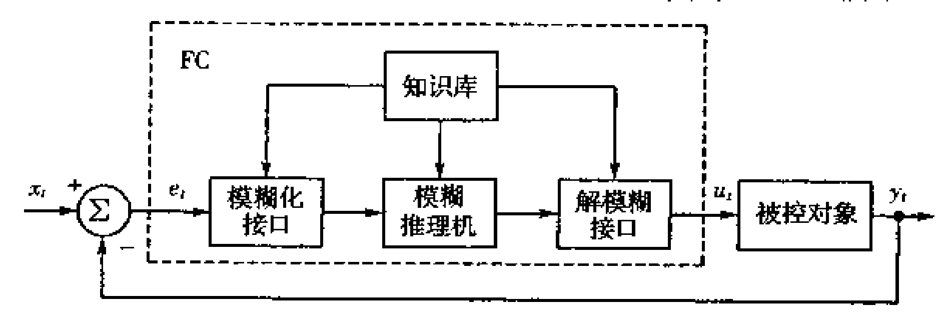


图 7.2 模糊控制器的结构

图中,u_t是被控对象的输入,y_t是被控对象的输出,x_t是参考输入,$e_t=x_t-y_t$是误差. 模糊控制器根据误差信号 e_t产生合适的控制作用 u_t,输出给被控对象. 模糊控制器主要由模糊化接口、知识库、模糊推理机、解模糊接口四部分组成,各部分的作用概述如下.

一、模糊化接口(Fuzzification)

模糊化接口接受的输入只有误差信号 e_t,由 e_t再生成误差变化率 $\dot{e}_t$,或误差的差分 Δe_t,模糊化接口主要完成以下两项功能.

(1)论域变换:e_t和 $\dot{e}_t$ 都是非模糊的普通变量,它们的论域(即变化范围)是实数域上的一个连续闭区间,称为真实论域,分别用 X 和 Y 来代表. 在模糊控制器中,真实论域要变换到内部论域 X'和 Y'. 如果内部论域是离散的(有限个元素),模糊控制器称为"离散论域的模糊控制器(D-FC)",如果内部论域是连续的(无穷多个元素),模糊控制器称为"连续论域的模糊控制器(C-FC)". 对于 D-FC,$X',Y'=\{0,\pm$整数$\}$;对于 C-FC,$X',Y'=[-1,1]$. 无论是 D-FC 还是 C-FC,论域变换后 e_t,$\dot{e}_t$ 变成 e_t^*,$\dot{e}_t^*$,相当乘了一个比例因子(还可能有偏移).

(2)模糊化:论域变换后 e_t^* 和 $\dot{e}_t^*$ 仍是非模糊的普通变量,对它们分别定义若干个模糊集合,如"负大(NL)"、"负中(NM)"、"负小(NS)"、"零(Z)"、"正小(PS)"、"正中(PM)"、"正大

(PL)"、…,并在其内部论域上规定各个模糊集合的隶属度函数.

在 t 时刻输入信号的值 e_t, $\dot{e}_t$,经论域变换后得到 e_t^*, $\dot{e}_t^*$,再根据隶属度函数的定义可以分别求出 e_t^*, $\dot{e}_t^*$ 对各模糊集合的隶属度,如 $\mu_{NL}(e_{t*})$、$\mu_{NM}(e_t^*)$、…,这样就把普通变量的值变成了模糊变量(即语言变量)的值,完成了模糊化的工作. 注意,在这里 e_t^*, $\dot{e}_t^*$ 既代表普通变量又代表模糊变量,作为普通变量时其值在论域 X', Y' 中,是普通数值,作为模糊变量时其值在论域[0,1]中,是隶属度.

二、知识库(Knowledge Base)

知识库中存贮着有关模糊控制器的一切知识,它们决定着模糊控制器的性能,是模糊控制器的核心. 知识库又分为两部分,分别介绍如下.

1. 数据库(Data Base)

数据库中存贮着有关模糊化、模糊推理、解模糊的一切知识,包括模糊化中的论域变换方法、输入变量各模糊集合的隶属度函数定义等,以及模糊推理算法、解模糊算法、输出变量各模糊集合的隶属度函数定义等.

2. 规则库(Rule Base)

就是模糊控制规则集,即以"if…then…"形式表示的模糊条件语句,如

R_1: if e_t^* is A_1 and $\dot{e}_t^*$ is B_1, then u_t^* is C_1,

R_2: if e_t^* is A_2 and $\dot{e}_t^*$ is B_2, then u^* is C_2,

……

……

R_n: if e_t^* is A_n and $\dot{e}_t^*$ is B_n, then u^* is C_n.

其中,e_t^* 和 $\dot{e}_t^*$ 就是前面所说的模糊语言变量,$A_1, A_2, \cdots, A_n$ 是 e_t^* 的模糊子集,$B_1, B_2, \cdots, B_n$ 是 $\dot{e}_t^*$ 的模糊子集,$C_1, C_2, \cdots, C_n$ 是 u^* 的模糊子集.

规则库中的 n 条规则是并列的,它们之间是"或"的逻辑关系,整个规则集的总模糊关系为

$$R = \bigcup_{i=1}^{n} R_i \tag{7.1.5}$$

三、模糊推理机(Inference Engine)

由 6.3.3 节介绍的模糊推理方法我们知道,模糊控制应用的是广义前向推理. 在 t 时刻,若输入量为 e_t^* 和 $\dot{e}_t^*$, $e_t^* \in X'$, $\dot{e}_t^* \in Y'$,且论域 X', Y' 和 Z' 都是离散的,e^* 在 X' 上对应矢量 $\boldsymbol{A}'$, $\dot{e}_t^*$ 在 Y' 上对应矢量 $\boldsymbol{B}'$,则推理结果是 Z' 上的矢量 $\boldsymbol{C}'$,

$$\boldsymbol{C}' = (A' \times \boldsymbol{B}') \cdot \boldsymbol{R} \tag{7.1.6}$$

四、解模糊接口(Defuzzification)

解模糊可以看作是模糊化的反过程,它要由模糊推理结果产生控制 u_t 的数值,作为模糊控制器的输出. 解模糊接口主要完成以下两项工作.

(1)解模糊:对 u_t 也要由真实论域 Z 变换到内部论域上 Z',对 $u_t^* \in Z'$ 定义若干个模糊子集,并规定各模糊子集的隶属度函数. 模糊推理是在内部论域上进行的,因此得到的推理结果 C' 是 Z' 上的模糊矢量,其元素为对 u^* 的某个模糊子集的隶属度. 对某组输入,一般会同时满足多条规则(称为激活),因此会有多个推理结果 C_i', i 为不同的模糊集合. 求 $\bigcup_i C_i'$,并用某种解模糊算法(见 7.1.1 节),可求得此时的内部控制量 u_t^*.

(2)论域反变换:得到的 $u_t^* \in Z'$,进行论域反变换即得到真正的输出 $u_t \in Z$,它仍然是非

模糊的普通变量.

§7.2 模糊控制器的种类和设计

7.2.1 模糊控制器的分类

一、D-FC和C-FC

模糊控制器输入输出变量的实际论域一般是实数域上的连续论域,而这些变量在模糊控制器内部的论域可以是连续的也可以是离散的,因此模糊控制器可以分成D-FC(离散论域的模糊控制器)和C-FC(连续论域的模糊控制器)两大类. D-FC控制时间短、性能稳定、对环境变化不敏感,而且算法非常简单、执行快,用八位的微型机即可实时控制,它适用于简单的、要求不高、不需要修改的场合,如家电. 而C-FC具有精度高、设计调试容易的优点,可用于多变量、高精度、需要修改的场合.

二、PD,PI,PID型的模糊控制器

PID(Proportional Integral Differential)控制是最常用的经典控制方法,控制作用 u 由偏差 e 的比例、积分、微分三项之和给出,

$$u = K_P e + K_I \int e dt + K_D \dot{e} \tag{7.2.1}$$

如果控制作用中只包括比例和积分两项,则为PI控制;如果控制作用只包含比例和微分两项,则为PD控制. PID控制中的比例增益 K_P、积分增益 K_I 和微分增益 K_D 均为常数,一旦控制器设计好以后,在控制过程中不再改变,因此PID控制属于线性定常数控制.

模糊控制本质上是非线性的,但是其输入变量也包括 e, $\dot{e}$ 和 $\int e dt$,输出变量也有 u(位量输出)和 Δu(增量输出)两种形式,与PID控制的输入输出变量是完全相同的. 类比传统的PD,PI,PID控制,可将模糊控制器分为以下三种类型.

1. PD型的模糊控制器

前面讲的模糊控制器(图7.2)输入是 e 和 $\dot{e}$,输出是 u,模糊控制器的功能可看做是一个非线性函数

$$u = f(e, \dot{e}) \tag{7.2.2}$$

这种模糊控制器的输入输出信号与PD控制器相同,控制特性也和PD控制器类似,故称为PD型的模糊控制器.

2. PI型的模糊控制器

如果输入仍为 e 和 $\dot{e}$,但输出改为控制的增量 $\Delta u \approx \dot{u} \Delta t$,则模糊控制器可表示为

$$\Delta u = f_1(e, \dot{e}) \tag{7.2.3}$$

上式两边对 t 积分可得

$$u = f_2\left(\int e dt, e\right) \tag{7.2.4}$$

可见此时的模糊控制器与PI控制器类似,故称为PI型的模糊控制器.

3. PID型的模糊控制器

同样的道理,PID型的模糊控制器有两种实现方法,即

$$u = f_1(e, \dot{e}, \int e\mathrm{d}t) \tag{7.2.5}$$

$$\Delta u = f_2(e, \dot{e}, \ddot{e}) \tag{7.2.6}$$

7.2.2　模糊控制器的设计方法

在设计模糊控制器时需要考虑以下几个方面.

(1)选择合适的模糊控制器类型.这个问题取决于被控对象的特性、控制要求和实现手段,需要具体问题具体分析.例如可采用最常用的连续论域、PI型、结论为数值的模糊控制器.

(2)确定输入输出变量的实际论域.一般来说输入信号 $e,\Delta e$ 和输出信号 Δu 的变化范围是由实际控制系统决定的,但有时也有一定的调整余地.调整 $e,\Delta e,\Delta u$ 的实际论域相当于改变比例因子,对控制器的性能影响很大.

(3)确定 $e,\Delta e,\Delta u$ 的模糊集个数及各模糊集的隶属度函数.模糊集的个数一般选为3、5、7个,个数多,模糊控制器的灵敏性好,但规则的数目成平方增长.

输出隶属度函数选为单点,可使解模糊简单.输入隶属度函数选为三角形或钟形(正态曲线)对控制性能影响不大,而选三角形时模糊化算法简单.

(4)设计模糊控制规则集.规则集是决定模糊控制器性能的关键因素,一般是根据经验来设计,或是用模糊聚类分析方法从控制器的已有输入输出样本中自动提取.

(5)选择模糊推理方法.按6.3.3选择合适的模糊推理方法.

(6)解模糊方法.按7.1.1节选择合适的解模糊判决方法.

§7.3　模糊控制的应用

7.3.1　蒸汽发动机的模糊控制系统

英国学者Mamdani和Assilian最早研究了小型实验室汽轮机的模糊控制系统,这一开创性的工作,为模糊控制理论和应用奠定了基础.

被控对象是蒸汽发动机和锅炉.蒸汽发动机是通过调整发动机汽缸的油门控制它的速度,而锅炉是以热量作为输入量,控制锅炉的气压.图7.3为蒸汽机和锅炉的模糊控制系统示意图.

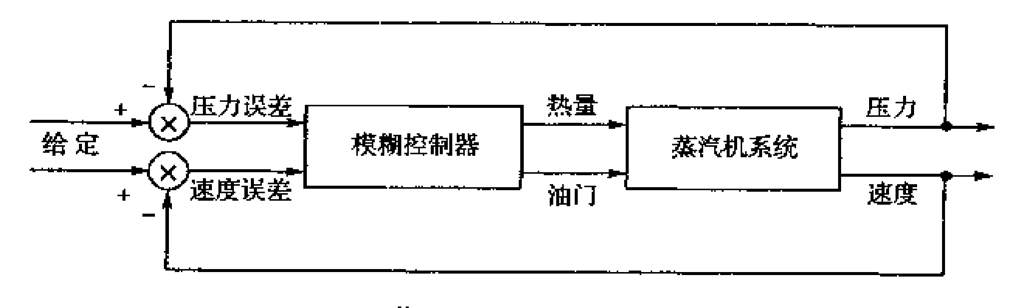


图7.3　蒸汽机和锅炉模糊控制系统

这是一个两输入两输出控制系统.控制变量分别为锅炉的热量与蒸汽机油门的开度.采用传统控制方法控制上述过程比较困难,原因在于这个过程的非线性、噪声以及两个控制回路间的强耦合.

一、模糊控制器的结构

模糊控制器采用以下6个模糊变量：

①PE(Pressure Error)——压力误差；

②SE(Speed Error)——速度误差；

③CPE(Change in Pressure Error)——压力误差的变化；

④CSE(Change in Speed Error)——速度误差的变化；

⑤HC(Heat Change)——热量变化；

⑥TC(Throttle Change)——油门变化.

其中PE,SE,CPE及CSE为输入模糊变量，而HC及TC为输出模糊变量.

模糊控制器是采取独立控制压力和速度的方式，因此，对于控制压力而言，它的输入变量为压力误差及压力误差的变化；而对于控制速度而言，它的输入变量为速度误差及速度误差的变化.

二、模糊变量的论域及其隶属度函数

把误差(PE,SE)论域量化为14挡，即{−6,−5,…,−1,−0,+0,+1,+2,…,+6}.

选择误差变量的模糊子集，即{PL(正大),PM(正中),PS(正小),PZ(正零),NZ(负零),NS(负小),NM(负中),NL(负大)}.

误差模糊变量的赋值如表7.1所示.

表7.1　误差模糊变量的赋值表

	−6	−5	−4	−3	−2	−1	−0	+0	+1	+2	+3	+4	+5	+6
PL	0	0	0	0	0	0	0	0	0	0	0.1	0.4	0.8	1.0
PM	0	0	0	0	0	0	0	0	0	0.2	0.7	1.0	0.7	0.2
PS	0	0	0	0	0	0	0	0.3	0.8	1.0	0.5	0.1	0	0
PZ	0	0	0	0	0	0	0	1.0	0.6	0.1	0	0	0	0
NZ	0	0	0	0	0.1	0.6	1.0	0	0	0	0	0	0	0
NS	0	0	0.1	0.5	1.0	0.8	0.3	0	0	0	0	0	0	0
NM	0.2	0.7	1.0	0.7	0.2	0	0	0	0	0	0	0	0	0
NL	1.0	0.8	0.4	0.1	0	0	0	0	0	0	0	0	0	0

误差模糊变量的赋值是根据表7.2中给出的模糊变量隶属度函数相应的表达式确定的.

表7.2　模糊变量的隶属度函数

模糊子集	表达式
PL	$1-\exp[-(0.5/\lvert 1-x\rvert)^{2.5}]$
PM	$1-\exp[-(0.25/\lvert 0.7-x\rvert)^{2.5}]$
PS	$1-\exp[-(0.25/\lvert 0.4-x\rvert)^{2.5}]$
PZ	$\exp[-5\lvert x-0.05\rvert]$
NZ	$\exp[-5\lvert x+0.05\rvert]$
NS	$1-\exp[-(0.25/\lvert -0.4-x\rvert)^{2.5}]$
NM	$1-\exp[-(0.25/\lvert -0.7-x\rvert)^{2.5}]$
NL	$1-\exp[-(0.5/\lvert -1-x\rvert)^{2.5}]$

将误差变化(CPS,CSE)论域量化为13挡，即{−6,−5,…,−1,0,+1,+2,…+6}.

选择误差变化的模糊子集，即{PL,PM,PS,Z,NS,NM,NL}.

误差变化的模糊变量赋值如表 7.3 所示.

表 7.3　误差变化模糊变量的赋值表

	−6	−5	−4	−3	−2	−1	+0	+1	+2	+3	+4	+5	+6
PL	0	0	0	0	0	0	0	0	0	0.1	0.4	0.8	1.0
PM	0	0	0	0	0	0	0	0	0.2	0.7	1.0	0.7	0.2
PS	0	0	0	0	0	0	0	0.9	1.0	0.7	0.2	0	0
Z	0	0	0	0	0	0.5	1.0	0.5	0	0	0	0	0
NS	0	0	0.2	0.7	1.0	0.9	0	0	0	0	0	0	0
NM	0.2	0.7	1.0	0.7	0.2	0	0	0	0	0	0	0	0
NL	1.0	0.8	0.4	0.1	0	0	0	0	0	0	0	0	0

热量变化(HC)的论域量化为 15 挡,即{−7,−6,…,−1,0,+1,+2,…+7}.

热量变化的模糊子集与误差变化的模糊子集相同,即{PL,PM,PS,Z,NS,NM,NL}.

热量变化的模糊变量赋值如表 7.4 所示.

表 7.4　热量变化模糊变量赋值表

	−7	−6	−5	−4	−3	−2	−1	0	+1	+2	+3	+4	+5	+6	+7
PL	0	0	0	0	0	0	0	0	0	0	0	0.1	0.4	0.8	1.0
PM	0	0	0	0	0	0	0	0	0	0.2	0.7	1.0	0.7	0.2	0
PS	0	0	0	0	0	0	0	0.4	1.0	0.8	0.4	0.1	0	0	0
Z	0	0	0	0	0	0	0.2	1.0	0.2	0	0	0	0	0	0
NS	0	0	0	0.1	0.4	0.8	1.0	0.4	0	0	0	0	0	0	0
NM	0	0.2	0.7	1.0	0.7	0.2	0	0	0	0	0	0	0	0	0
NL	1.0	0.8	0.4	0.1	0	0	0	0	0	0	0	0	0	0	0

油门变化(TC)的论域量化为 5 挡,即{−2,−1,0,+1,+2}.

油门变化的模糊子集选为{PL,PS,Z,NS,NL}.

油门变化的模糊变量赋值由表 7.5 给出.

表 7.5　油门变化的模糊变量赋值表

	−2	−1	0	+1	+2
PL	0	0	0	0.5	1.0
PS	0	0	0.5	1.0	0.5
Z	0	1.0	0.5	0	0
NS	0.5	1.0	0.5	0	0
NL	1.0	0.5	0	0	0

三、模糊控制规则

两个控制环分别制定两套模糊控制规则如下:

1. 压力控制规则

　　if PE =NL then if CPE =not(NL or NM) then HC=PL

or　if PE = (NL or NM)then if CPE=NS then NC= PM

or　if PE = NS then if CPE=(PS or NZ) then HC=PM

or　if PE=NZ then if CPE=(PL or PM) then HC=PM

or　if PE=NZ then if CPE=(NL or NM) then HC=NM

or　if PE=(PZ or NZ) then if CPE=NZ then HC=NZ

or　if PE=PZ then CPE=(NL or NM) then HC=PM

or　if PE=PZ then if CPE=(PL or PM) then HC=NM

or　if PE=PS then if CPE=(PS or NZ) then HC=NM

or　if PE= PL or PM) then if CPE=NS then HC=NM

or　if PE=PL then if CPE=not (NL or NM) then HC=NL

or　if PE=NZ then if CPE=PS then HC=PS

or　if PE=NZ then if CPE=NS then HC=NS

or　if PE=PZ then if CPE=NS then HC=PS

or　if PE=PZ then if CPE=PS then HC=NS

2. 速度控制规则

　　if SE = NL then if CSE=not(NL or NM)then TC=PL

or　if SE=NM then if CSE=(PL or PM or PS)then TC=PS

or　if SE=NS then if CSE=(PL or PM)then TC=PS

or　if SE=NZ then if CSE=PL then TC=PS

or　if SE=(PZ or NZ)then if CSE=(PS or NS or NZ)then TC=NZ

or　if SE=PZ then if CSE=PL then TC=NS

or　if SE=PS then if CSE=(PL or PM)then TC=NS

or　if SE=PM then if CSE=(PL or PM or PS)then TC=NS

or　if SE=PL then if CSE=not(NL or NM)then TC=NL

四、模糊控制的结果

Mamdani 比较了锅炉出口压力的模糊控制结果和传统的控制系统 DDC 控制的结果，二种控制系统都是调整到最佳状态. 两种控制结果的阶跃响应曲线如图 7.4 所示.

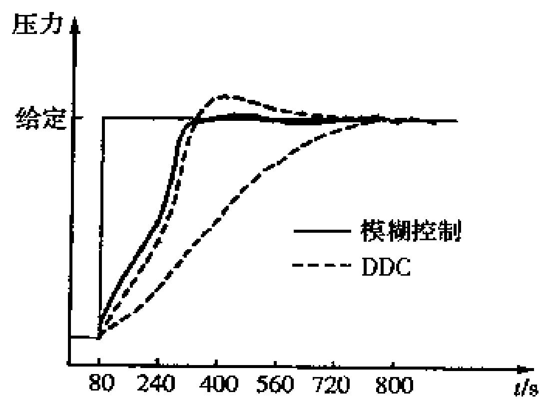


图 7.4　阶跃响应曲线

实验表明，当工作条件在大范围内变动时，DDC 难以调整，而且由于动力过程的改变，DDC 控制方式无法在同一组参数下得到满意的控制特性. 与此相反，模糊控制系统由于对过程参数的变化很不灵敏，所以它在所有的工作点都能获得很好的控制效果. 由于蒸汽机的特性随时间非线性变化，因而 DDC 控制应该经常进行参数的再调整. 模糊控制器却没有这种必要，因为它具有较强的适应能力和抑制噪声能力.

7.3.2　还原炉温度的模糊控制系统

九管还原炉是对氧化钨粉末加热并通入氢气使其还原成钨的设备，分为六个温区，要求每

个温区的温度稳定在某个给定值，如第一温区 550℃，第二温区 650℃，…，偏差不大于±5℃. 因为有六个温区，故对每个温区设计了一个温度控制系统，如图 7.5 所示.

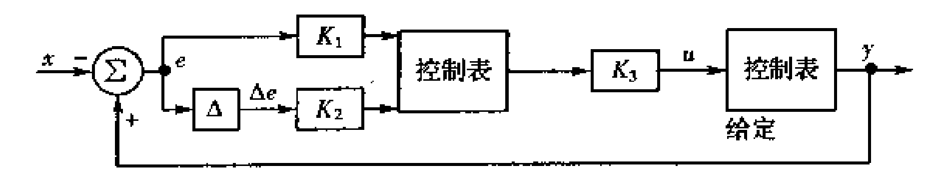


图 7.5　还原炉温度的模糊控制系统

图中 x 是温度给定值，y 为被控对象的温度量测值，$e=y-x$ 是温度误差，Δe 是温度误差差分，$\Delta e_t=e_{t-1}-e_t$，u 为控制量，它对应着电加热装置的可控硅导通角的变化量.

一、模糊变量的论域及其隶属度函数

$e,\Delta e,u$ 都是实数域上的连续变量，变化范围分别为：$e\in[-30,30]$，$\Delta e\in[-24,24]$，$u\in[-36,36]$.

将温度误差 e 量化为 14 挡，即 $e^*\in\{-6,-5,\cdots,-1,-0,+0,+1,+2,\cdots+6\}$.

选择温度误差 e^* 的模糊子集，即 $E_i=\{$PL(正大)，PM(正中)，PS(正小)，PZ(正零)，NZ(负零)，NS(负小)，NM(负中)，NL(负大)$\}$，$i=1,2,\cdots,8$.

温度误差 e^* 的模糊变量的赋值如表 7.6 所示.

表 7.6　温度误差 e^* 的模糊变量赋值表

	−6	−5	−4	−3	−2	−1	−0	+0	+1	+2	+3	+4	+5	+6
PL	0	0	0	0	0	0	0	0	0	0	0	0.2	0.7	1.0
PM	0	0	0	0	0	0	0	0	0	0.2	0.7	1.0	0.7	0.2
PS	0	0	0	0	0	0	0	0.1	0.7	1.0	0.7	0.1	0	0
PZ	0	0	0	0	0	0	0	1.0	0.7	0.1	0	0	0	0
NZ	0	0	0	0	0.1	0.7	1.0	0	0	0	0	0	0	0
NS	0	0	0.1	0.7	1.0	0.7	0.1	0	0	0	0	0	0	0
NM	0.2	0.7	1.0	0.7	0.2	0	0	0	0	0	0	0	0	0
NL	1.0	0.7	0.2	0	0	0	0	0	0	0	0	0	0	0

将温度误差差分 Δe 量化为 13 挡，即 $\Delta e^*\in\{-6,-5,\cdots,-1,0,+1,+2,\cdots+6\}$.

选择温度差分 Δe^* 的模糊子集，即 $\Delta E_j=\{$PL,PM,PS,Z,NS,NM,NL$\}$，$j=1,2,\cdots,7$.

温度差分 Δe^* 的模糊变量的赋值如表 7.7 所示.

表 7.7　温度差分 Δe^* 的模糊变量赋值表

	−6	−5	−4	−3	−2	−1	0	+1	+2	+3	+4	+5	+6
PL	0	0	0	0	0	0	0	0	0	0	0.2	0.7	1.0
PM	0	0	0	0	0	0	0	0	0.2	0.8	1.0	0.8	0.2
PS	0	0	0	0	0	0	0	0.8	1.0	0.8	0.2	0	0
Z	0	0	0	0	0	0.5	1.0	0.5	0	0	0	0	0
NS	0	0	0.2	0.8	1.0	0.8	0	0	0	0	0	0	0
NM	0.2	0.8	1.0	0.8	0.2	0	0	0	0	0	0	0	0
NL	1.0	0.7	0.2	0	0	0	0	0	0	0	0	0	0

将控制量 u 量化为 13 挡，即 $u^*\in\{-6,-5,\cdots,-1,0,+1,+2,\cdots+6\}$.

选择控制量 u^* 的模糊子集，即 $U_k=\{PL,PM,PS,Z,NS,NM,NL\}$，$k=1,2,\cdots,7$.

控制量 u^* 的模糊变量的赋值如表 7.8 所示.

表 7.8 控制量 u^* 的模糊变量赋值表

	−6	−5	−4	−3	−2	−1	0	+1	+2	+3	+4	+5	+6
PL	0	0	0	0	0	0	0	0	0	0	0.2	0.7	1.0
PM	0	0	0	0	0	0	0	0	0.2	0.8	1.0	0.8	0.2
PS	0	0	0	0	0	0	0.1	0.8	1.0	0.8	0.1	0	0
Z	0	0	0	0	0	0.5	1.0	0.5	0	0	0	0	0
NS	0	0	0.1	0.8	1.0	0.8	0.1	0	0	0	0	0	0
NM	0.2	0.8	1.0	0.8	0.2	0	0	0	0	0	0	0	0
NL	1.0	0.7	0.2	0	0	0	0	0	0	0	0	0	0

二、模糊控制规则

总结操作人员的经验给出 52 条控制规则，归纳于表 7.9. 控制规则的一般式是

$$\text{if } e^*=E_i \text{ and } \Delta e^*=\Delta E_j \text{, then } u^*=U_k$$

$$i=1,2,\cdots,8;j=1,2,\cdots,7;k=1,2,\cdots,7$$

表 7.9　控制规则表

u^*		e^*							
		NL	NM	NS	NZ	PZ	PS	PM	PL
Δe^*	PL	PL	PM	NM	NM	NM	NL	NL	
	PM	PL	PM	NM	NM	NM	NS	NS	
	PS	PL	PM	NS	NS	NS	NS	NM	NL
	Z	PL	PM	PS	Z	Z	NS	NM	NL
	NS	PL	PM	PS	PS	PS	PS	NM	NL
	NM		PL	PS	PS	PM	PM	NM	NL
	NL		PL	PL	PM	PM	PM	NM	NL

三、求模糊关系矩阵 $\boldsymbol{R}$ 和模糊控制表

由于论域是离散的，模糊控制规则集可以表示为一个模糊关系矩阵 $\boldsymbol{R}$：

$$\boldsymbol{R}=\bigcup_{\forall i,j,k}(\boldsymbol{E}_i\times\Delta\boldsymbol{E}_j)\times\boldsymbol{U}_k \tag{7.3.1}$$

可见，$\boldsymbol{R}$ 是一个 $(14\times13)\times13$ 的大矩阵，占 2366 个内存单元.

算出 $\boldsymbol{R}$ 以后，我们可以对 e^* 和 Δe^* 设不同的值，例如 $\Delta e^*=-6$，$\Delta e^*=-6$，将它们看成是模糊单点，这时的输入是模糊矢量 $\boldsymbol{E}'=[1,0,\cdots,0]^{\mathrm{T}}$，可求出输出模糊矢量 $\boldsymbol{U}'$：

$$\boldsymbol{U}'=(\boldsymbol{E}'\times\Delta\boldsymbol{E}')\cdot\boldsymbol{R} \tag{7.3.2}$$

这里 $(\boldsymbol{E}'\times\Delta\boldsymbol{E}')$ 是一个 14×13 的矩阵，将其按行排成矢量，再去与 $\boldsymbol{R}$ 合成. 由于 $(\boldsymbol{E}'\times\Delta\boldsymbol{E}')$ 此时只有第一个元素为 1，其他元素皆为 0，$\boldsymbol{U}'$ 即是 $\boldsymbol{R}$ 的第一行. $\boldsymbol{U}'$ 中隶属度最大对应的 u^* 即为此时的输出. 用同样的方法，对每对输入 e_i^*，Δe_j^* 都可求出相应的输出 u_{ij}^*，将它们整理成模糊控制表，如表 7.10 所示.

表 7.10 模糊控制查询表

u^*		e^*													
		−6	−5	−4	−3	−2	−1	−0	+0	+1	+2	+3	+4	+5	+6
Δe^*	−6	+6	+5	+6	+5	+3	+3	+3	+3	+2	+1	0	0	0	0
	−5	+5	+5	+5	+5	+3	+3	+3	+3	+2	+1	0	0	0	0
	−4	+6	+5	+6	+5	+3	+3	+3	+3	+2	+1	0	0	0	0
	−3	+5	+5	+5	+5	+4	+4	+4	+4	+2	−1	−1	−1	−1	−1
	−2	+6	+5	+6	+5	+3	+3	+1	1	0	0	−2	−3	−3	−3
	−1	+6	+5	+6	+5	+3	+3	+1	0	0	−2	−2	−3	−3	−3
	0	+6	+5	+6	+5	+3	+1	0	0	−1	−3	−5	−6	−5	−6
	+1	+3	+3	+3	+2	0	0	0	−1	−3	−3	−5	−6	−5	−6
	+2	+3	+3	+3	+1	0	0	−1	−1	−3	−3	−5	−6	−5	−6
	+3	+1	+1	+1	0	0	0—	1—	1—	2—	2—	5—	5—	5—	5
	+4	0	0	0	−1	−1	−2	−3	−3	−3	−3	−5	−6	−5	−6
	+5	0	0	0	−1	−1	−2	−3	−3	−3	−3	−5	−5	−5	−5
	+6	0	0	0	−1	−1	−1	−3	−3	−3	−3	−5	−5	−5	−6

四、实时控制

控制表可离线算出，在实时控制时得到 e_t 和 Δe_t 后，先变换成离散量 e_t^* 和 Δe_t^*，查表得到 u_t^*，再求出真正的连续控制量 u_t.

以上设计的模糊控制器已成功地用于还原炉的温度控制，取得了很好的控制效果. 温度控制的上升时间短，超调量小，控制性能稳定，对环境变化不敏感，而且算法非常简单，执行快，用八位的微型机即可实时控制.

§7.4　模糊控制规则的调整

由于模糊控制理论仍不完备，加上模糊系统的复杂性，人们总结不出完整的经验来，这样就使得模糊控制有时显得很粗糙. 另外，即使模糊规则很完善，但由于过程在不断变化，总是按原来的规则进行控制，所得结果可能也与实际不符. 因此人们设计出这样的控制器，它能在运行中自动地修改、完善和调整控制规则，使系统性能不断改善，从而达到预期的效果.

7.4.1　带有修正因子的模糊控制器

设 e^* 为误差，$\dot{e}^*$ 为误差变化率，u^* 是控制量. 它们的模糊变量均可取为{PL(正大)，PM(正中)，PS(正小)，Z(零)，NS(负小)，NM(负中)，NL(负大)}，并且用数值分级定义为{+3，+2，+1，0，−1，−2，−3}.

于是，一个最简单的控制表可用表 7.11 来概括.

表 7.11 最简单的控制规则表

u^*		e^*						
		−3	−2	−1	0	1	2	3
$\dot{e}^*$	−3	−3	−3	−2	−2	−1	−1	0
	−2	−3	−2	−2	−1	−1	0	1
	−1	−2	−2	−1	−1	0	1	1
	0	−2	−1	−1	0	1	1	2
	1	−1	−1	0	1	1	2	2
	2	−1	0	1	1	2	2	2
	3	0	1	1	2	2	3	3

表 7.11 构造规则可用下式表达：

$$u^* = \left\langle \frac{e^* + \dot{e}^*}{2} \right\rangle \tag{7.4.1}$$

设 $\langle a \rangle$ 表示一个与 α 同号而其绝对值大于或等于 $|\alpha|$ 的最小整数. 例如：$\langle 0 \rangle = 0$，$\langle 0.5 \rangle = 1$，$\langle -0.5 \rangle = -1$. 现在，给出带修正因子的控制规则：

$$u^* = \langle \alpha e^* + (1-\alpha)\dot{e}^* \rangle \tag{7.4.2}$$

其中 α 在[0,1]区间取值. 当 $\alpha=0.5$ 时得到的控制规则表就是表 7.11.

下面给出当 $\alpha=0.7$ 时的控制规则表 7.12 与 $\alpha=0.2$ 时控制规则表 7.13.

表 7.12 $\alpha=0.7$ 时的控制规则表

u^*		e^*						
		−3	−2	−1	0	1	2	3
$\dot{e}^*$	−3	−3	−3	−2	−2	−2	−2	−1
	−2	−2	−2	−2	−1	−1	−1	−1
	−1	−2	−1	−1	−1	0	0	0
	0	−1	−1	0	0	0	1	1
	1	0	0	0	1	1	1	2
	2	1	1	1	1	2	2	2
	3	1	2	2	2	2	3	3

表 7.13 $\alpha=0.2$ 时的控制规则表

u^*		e^*						
		3	−3	−2	−1	0	1	2
$\dot{e}^*$	−3	−3	−2	−1	−1	0	1	2
	−2	−3	−2	−1	0	0	1	2
	−1	−3	−2	−1	0	1	1	2
	0	−2	−2	−1	0	1	2	2
	1	−2	−1	−1	0	1	2	3
	2	−2	−1	0	0	1	2	3
	3	−1	−1	0	1	1	2	3

通过表 7.11，表 7.12，表 7.13，可以看出，通过调整系数 α，可以得到不同的控制规则，用 α 作为调解误差及误差变化率的权重是很符合人们进行控制活动的思维特点的.

这种方法虽然能调整控制规则，但 α 一经确定，误差及误差变化率的权重比例也就确定了. 可是控制系统在不同的状态下，对误差与误差变化率的权重要求是不同的. 当误差较大时，

控制系统的主要任务是消除误差，这时误差的权重应占较大比例. 反之当误差较小时，为了使系统尽快稳定，则误差变化率占主要地位，应增大其权重. 基于上述想法，人们提出了在不同的状态下用因子调整控制规则. 控制规则可由下二式给出：

$$u^* = \begin{cases} \langle \alpha_1 e^* + (1-\alpha_1)\dot{e}^* \rangle, & 当\ e_i^* = \pm 1, 0 \\ \langle \alpha_2 e^* + (1-\alpha_2)\dot{e}^* \rangle, & 当\ e^* = \pm 2, \pm 3 \end{cases} \tag{7.4.3}$$

式中 $1>\alpha_2\geqslant\alpha_1>0$.

$$u^* = \begin{cases} \langle \alpha_0 e_i^* + (1-\alpha_0)\dot{e}^* \rangle, & 当\ e^* = 0 \\ \langle \alpha_1 e^* + (1-\alpha_1)\dot{e}^* \rangle, & 当\ e^* = \pm 1 \\ \langle \alpha_2 e^* + (1-\alpha_2)\dot{e}^* \rangle, & 当\ e^* = \pm 2 \\ \langle \alpha_3 e^* + (1-\alpha_3)\dot{e}^* \rangle, & 当\ e^* = \pm 3 \end{cases} \tag{7.4.4}$$

式中 $1>\alpha_3\geqslant\alpha_2\geqslant\alpha_1\geqslant\alpha_0>0$.

7.4.2　自适应模糊控制器

人们希望系统能在运行中自动地修改、调整和完善模糊控制规则，从而使系统的性能不断改善，直到系统的输出特性达到预定的精度为止，这就是所谓的自适应模糊控制系统.

自适应模糊控制器是在简单模糊控制器的基础上，增加了三个功能块而构成的一种模糊控制器，其结构如图 7.6 所示. 图中上面的虚线框内的三个功能块即为增加的部分，它们分别是：

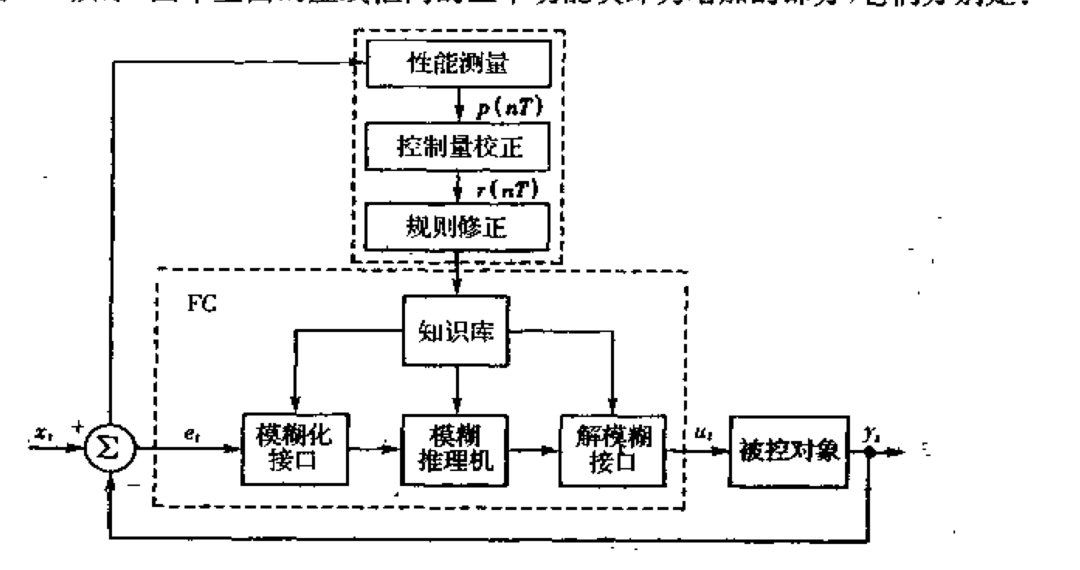


图 7.6　自适应模糊控制器的结构

一、性能测量

用于测量实际输出特性与希望特性的偏差，以便为控制规则的修正提供信息，即确定输出响应的校正量 P.

在模糊控制器中（一般指二维），通常选“偏差”和“偏差变化率”作为两个参量，用以衡量输出特性的偏离情况. 因此，可以根据这两个参数的采样值 $e(nT)$ 和 $\dot{e}(nT)$ 计算出对输出特性所需要的校正量 $p(nT)$，并且可以采用模糊集合论的方法总结出一套性能测量规则.

类似于简单模糊控制器的控制表，校正表也可以根据性能测量规则来确定. 例如，将“偏差”、“偏差变化率”和“校正量”的分级定义如下：

$e(nT)$:{$-6,-5,\cdots,-1,-0,+0,+1,\cdots,+5,+6$}

$\dot{e}(nT)$:{$-6,-5,\cdots,-1,0,+1,\cdots,+5,+6$}

$p(nT)$:{$-6,-5,\cdots,-1,0,+1,\cdots,+5,+6$}

表 7.14 是一个性能校正表. 例如,表中第一行偏差为负,且在最大(－6)级上,当偏差变化率是朝向设定点时,其发展趋势是逐渐减少,因此不必加校正量;当偏差变化率是离开设定点时,均施加(＋6)级的校正量,抑制偏差的进一步变化并使其迅速减小,以便使系统输出特性向期望的方向变化.

表 7.14　性能测量校正表

		趋向设定点						离开设定点						
		−6	−5	−4	−3	−2	−1	0	+1	+2	+3	+4	+5	+6
负偏差	−6	0	0	0	0	0	0	6	6	6	6	6	6	6
	−5	0	0	0	2	2	3	6	6	6	6	6	6	6
	−4	0	0	0	2	4	5	6	6	6	6	6	6	6
	−3	0	0	0	2	2	3	4	4	4	4	5	5	6
	−2	0	0	0	0	0	0	2	2	2	3	4	5	6
	−1	0	0	0	0	0	0	1	1	1	2	3	4	5
	−0	0	0	0	0	0	0	0	0	0	1	2	3	4
正偏差	+0	0	0	0	0	0	0	0	0	0	−1	−2	−3	−4
	+1	0	0	0	0	0	0	−1	−1	−1	−2	−3	−4	−5
	+2	0	0	0	0	0	0	−2	−2	−2	−3	−4	−5	−6
	+3	0	0	0	−2	−2	−3	−4	−4	−4	−4	−5	−5	−6
	+4	0	0	0	−2	−4	−5	−6	−6	−6	−6	−6	−6	−6
	+5	0	0	0	−2	−2	−3	−6	−6	−6	−6	−6	−6	−6
	+6	0	0	0	0	0	0	−6	−6	−6	−6	−6	−6	−6

从表 7.14 中可以看出,离开设定点越远,需要的输出校正就越大. 由于设定点是正、负对称的,所以负偏差时输出校正需增加,反之输出校正需减小.

如果用 Π 表示校正,那么表 7.14 可用下式表示,即

$$p(nT) = \Pi[e(nT),\dot{e}(nT)] \tag{7.4.5}$$

二、控制量的校正

得到了输出性能的校正量 $p(nT)$,还必须利用它计算出对被控对象的输入量的校正量 $r(nT)$才能使下一次的输出特性得以改变,这牵涉到系统的输入与输出之间关系,例如一个两输入两输出的系统,设过程的状态方程为

$$\dot{X} = F(X,U,V) \tag{7.4.6}$$

$$\dot{Y} = G(Y,U,V) \tag{7.4.7}$$

其中 U,V 为输入,对于较小的输入变化有

$$\begin{bmatrix}\delta\dot{X}\\ \delta\dot{Y}\end{bmatrix}=\begin{bmatrix}\frac{\delta F}{\partial U} & \frac{\partial F}{\partial V}\\ \frac{\partial G}{\partial U} & \frac{\partial G}{\partial V}\end{bmatrix}\begin{bmatrix}\partial U\\ \partial V\end{bmatrix} \tag{7.4.8}$$

如果输入变化是 $\Delta U,\Delta V$,则经过一个采样周期 T 后,引起输出变化为

$$\begin{bmatrix}\Delta X\\ \Delta Y\end{bmatrix}\approx\begin{bmatrix}T\delta\dot{X}\\ T\delta\dot{Y}\end{bmatrix}=TJ\begin{bmatrix}\Delta U\\ \Delta V\end{bmatrix}=M\begin{bmatrix}\Delta U\\ \Delta V\end{bmatrix} \tag{7.4.9}$$

其中 J 是雅可比矩阵，上式确定了输入变化 ΔU、ΔV 与输出变化 ΔX、ΔY 之间关系的一个增量模型，可把矩阵 M 称为过程的增量模型. 如果要求的输出校正是 $p_1(nT)$、$p_2(nT)$，那么必须有输入量校正 $r_1(nT)$，$r_2(nT)$，它们可由上面的增量模型求得如下

$$\begin{bmatrix} r_1(nT) \\ r_2(nT) \end{bmatrix} = M^{-1} \begin{bmatrix} p_1(nT) \\ p_2(nT) \end{bmatrix} \tag{7.4.10}$$

三、控制规则的校正

对控制量的校正通过修改控制规则来实现. 控制规则的校正过程是根据所需要的输入校正量 $r(nT)$ 来进行的. 现在以单输入单输出过程加以说明.

设系统有一定的滞后，设 $e(nT-mT)$、$\dot{e}(nT-mT)$、$u(nT-mT)$ 分别表示原来的偏差、偏差变化率和控制量. 而要求校正后控制量是 $u(nT-mT)+r(nT)$，将这些量进行模糊化：

$$\begin{aligned} \boldsymbol{E}(nT-mT) &= F[e(nT-mT)] \\ \boldsymbol{C}(nT-mT) &= F[\dot{e}(nT-mT)] \\ \boldsymbol{U}(nT-mT) &= F[u(nT-mT)] \\ \boldsymbol{V}(nT-mT) &= F[u(nT-mT)+r(nt)] \end{aligned} \tag{7.4.11}$$

原来的控制规则为

$$\begin{aligned} &\text{if} \quad e=\boldsymbol{E}(nT-mT) \\ &\text{and} \quad \dot{e}=\boldsymbol{C}(nT-mT) \\ &\text{then} \quad u=\boldsymbol{U}(nT-mT) \end{aligned}$$

现在的控制规则为

$$\begin{aligned} &\text{if} \quad e=\boldsymbol{E}(nT-mT) \\ &\text{and} \quad \dot{e}=\boldsymbol{C}(nT-mT) \\ &\text{then} \quad \text{u}=\boldsymbol{V}(nT-mT) \end{aligned}$$

将上面两式写成关系矩阵为

$$\begin{aligned} \boldsymbol{R}'(nT) &= \boldsymbol{E}(nT-mT) \times \boldsymbol{C}(nT-mT) \times \boldsymbol{U}(nT-mT) \\ \boldsymbol{R}''(nT) &= \boldsymbol{E}(nT-mT) \times \boldsymbol{C}(nT-mT) \times V(nT-mT) \end{aligned} \tag{7.4.12}$$

用语句形式表示新的校正关系矩阵为

$$\boldsymbol{R}(nT+T) = \{\boldsymbol{R}(nT) \text{ 但不是 } \boldsymbol{R}'(nT)\}, \text{否则 } \boldsymbol{R}''(nT) \tag{7.4.13}$$

其中，$\boldsymbol{R}(nT)$ 是现在的控制器关系矩阵，$\boldsymbol{R}(nT+T)$ 是校正后控制器新的关系矩阵. 上式也可写成：

$$\boldsymbol{R}(nT+T) = \{\boldsymbol{R}(nT) \wedge \overline{\boldsymbol{R}'(nT)}\} \vee \boldsymbol{R}''(nT) \tag{7.4.14}$$

这就是构成修正控制器的一般方法. 由它就可以根据测得的偏差 $e(nT)$、偏差变化率 $\dot{e}(nT)$ 与 $\boldsymbol{R}(nT+T)$ 合成，求得控制量模糊集. 然后经过解模糊决策后，就可得到确定的控制量，加到系统中去，完成了一个控制动作.

第八章　模糊神经网络与神经模糊系统

§8.1　神经网络与模糊系统

8.1.1　神经网络与模糊系统的结合是发展的必然

人脑实际上是神经网络与模糊系统有机的自然结合. 从物质结构上看，人脑是一种典型的生物神经元网络，人脑的思维活动和智能行为是大量的神经元协同运作的宏观表现；而从主要的功能角度看，人脑又是个模糊系统. 人工神经网络是对人脑的结构和工作方式的近似和简化；而人工模糊系统是以模糊数学为基础的，在较高层次上对人脑思维的模糊性方面进行工程化的模拟.

神经网络和模糊系统是当前两种主要的智能技术，它们都能模拟人的智能行为，不需要精确的数学模型，能够解决传统技术无法解决的许多复杂的、不确定性的、非线性的问题，而且易于用硬件或软件来实现. 神经网络和模糊系统又具有各自的特点. 模糊系统能够直接地表示逻辑，适于直接的或高级的知识表达，擅长利用人的经验性知识，但是模糊规则的自动提取和模糊变量隶属度函数的自动生成和优化，一直是困扰模糊信息处理技术进一步推广的难题. 现有模糊系统方法缺乏使系统具备学习能力的手段，而这正是神经网络的优势所在. 另一方面，神经网络虽然可以通过学习获得用数据表达的知识，但这种知识在神经网络中是用权值、阈值的形式隐含表达的，因面难以理解.

随着对神经网络和模糊系统研究的深人，两个领域间相互独立的关系逐渐改变. 一个领域的缺乏可以通过另一个领域的优势来补偿，作到优势互补. 将模糊技术引入神经两络，可以大大拓宽神经网络的处理信息的范围和能力，使其不仅能处理精确信息，也能处理模糊信息，不仅能实现精确性联想映射，也可以实现不精确性联想及映射，即模糊联想映射. 采用神经网络来进行模糊信息处理，则使得模糊规则的自动提取和隶属度函数的自动优化有可能得以解决，从而使模糊系统成为一种自适应的系统. 将神经网络和模糊系统进行结合，即将符号逻辑推理方法与联接机制方法进行结合，将数值逻辑方法和模糊逻辑方法进行结合，其优势将远远高于单项研究.

通过比较神经网络与模糊系统可以发现，由于神经元的阈值函数为S型，使神经元的输出为[0,1]区域上的连续值，它与模糊系统的隶属度函数是相似的. 另外，在模糊推理中，规则前提部分的极小运算相当于神经元输人信号与加权系数的乘积，由推理规则的结论部分获得最后推理值的极大运算相当于神经元内的输人信号求和. 这些正是神经网络与模糊系统相结合的理论基础.

神经网络与模糊系统集成的系统，将兼有二者之长，既具有神经网络的学习能力、优化能力和连接式结构，又具有模糊系统的类似于人类思维方式的 if-then 规则，并易于嵌人专家知识. 二者的结合是发展的必然.

8.1.2 神经网络与模糊系统的结合方式

将神经网络与模糊系统相结合，可以将两者的优势集成在一起，使得神经网络或模糊系统在保持原有功能的基础上扩充其功能，既具有神经网络的准确拟合能力和学习能力，也能像模糊模型那样，易于被人们所理解. 人们可以用模糊模型将专家知识转换到神经网络中，然后，这个神经网络应用于一个目标系统，在运行中通过学习，来提高模型的准确性. 经过学习之后，就可以将神经网络的参数转换到模糊模型中，帮助解释神经网络中存储的知识. 在开始时，目标系统的模型是基于模糊规则而构造的，比较粗糙，经过这一个合作系统的学习后，目标系统的模型就可以得到进一步改善.

神经网络与模糊系统的合作系统的结构图如图 8.1 所示.

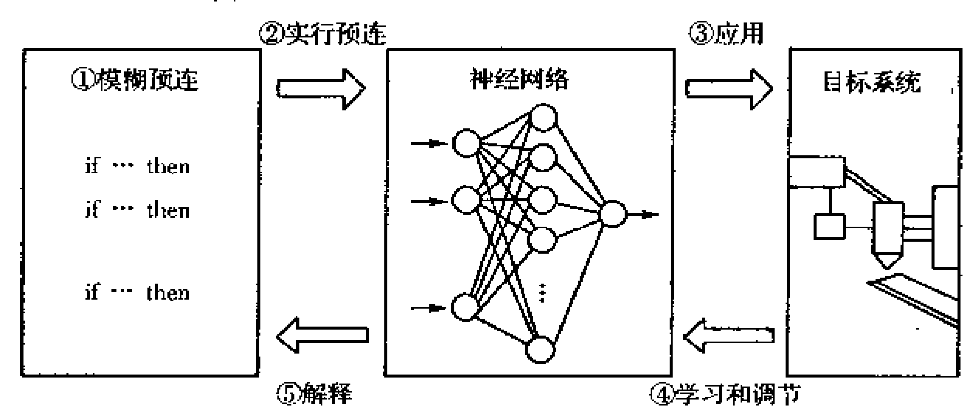


图 8.1 神经网络与模糊系统的合作系统的结构图

这个系统的构造和工作过程简述如下：

(1)根据专家对目标系统的知识，得到模糊系统的隶属度和模糊规则，从而可以构造出一个模糊系统；

(2)根据模糊系统确定神经网络的连接方式与所有的连接权值；

(3)将已经确定的神经网络应用于实际的目标系统；

(4)从目标系统的传感器取得数据，利用所得数据对神经网络进行训练、学习，以提高它的准确性；

(5)将经过学习后的神经网络的连接权值的变化加以解释，从而增强对神经网络内部过程的理解.

神经网络与模糊系统的结合大体上分为以下两种方式：

(1)模糊神经网络：这是模糊系统向神经网络的结合，把模糊逻辑插入到神经网络中，使神经网络具有逻辑推理功能，利用模糊逻辑提高神经网络的学习速度. 模糊神经网络保留了神经网络的基本性质与结构，只是将其中某些元件"模糊化"，其本质上还是神经网络，故它主要应用于模式识别领域.

(2)神经模糊系统：这是神经网络向模糊系统的结合，把神经网络的学习功能赋予模糊系统，使模糊系统能自动从学习中获取模糊规则. 在神经模糊系统中，神经网络用于对模糊集合扩大的数值进行处理，譬如隶属度函数的选取和模糊集合(用作模糊规则)之间映射实现. 因为神经模糊系统本质上还是模糊逻辑系统，所以它主要用在控制领域.

将神经网络与模糊系统相结合的尝试主要有以下几个方面：模糊聚类网络、基于模糊推理的神

经网络、自适应模糊控制系统、适应性模糊联想记忆系统和基于神经网络的模糊系统建模等.

§8.2　模糊神经网络

把模糊逻辑技术引入神经网络形成的系统称为模糊神经网络(Fuzzy Neural Network,简称为 FNN). FNN 事实上是模糊化的神经网络,即神经网络的某些元件(输入、输出、转移函数、权值、学习算法等)被模糊化.

8.2.1　模糊神经网络分类器

传统的多层感知器用作分类器时,某一数据只能属于某一类,而不能同时属于几类. 模糊神经网络就是将模糊的概念结合于网络的各层中,它的输入和输出都是具有语义性质的隶属度,可以利用基于梯度下降原理的逆向传播算法来修改其连接权值. 在应用中,分为有导师的学习和未知数据的分类两个阶段.

下例是军事上应用两层网络来分出坦克车与装甲车. 结构如图 8.2 所示. 22 个输入分别表示 22 个特征,其中 11 个是用电视播像机摄得的图像,为外形特征;另外 11 个为红外图像特征. 所有特征中,有 8 个是统计特征,其他是句法特征. 三个输出分别表示坦克车、装甲车和假目标. 通过 γ 模型的学习,从 22 个特征中只选出了 14 个特征,其他都被简化了,而且每次学习都采用不同的 75% 的样本,其他 25% 样本作为测试,其结果是相同的. 这种简化后的网络其识别率仍然很高.

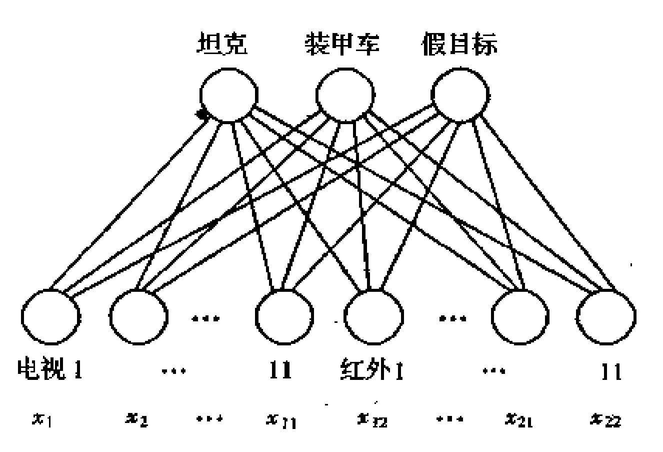


图 8.2　两层分类网络

如果把两层改为多层,学习结果如图 8.3 所示,经过 γ 模型学习后,网络层次分明,原有的 22 个输入神经元,经过多层学习后,只剩下 14 个,与两层网络的结果相似,其中 8 个为电视的图像特征,另外 6 个为红外图像特征. 而第二层神经元变成了 8 个,每个神经元只联系一类,如第二层中第一个神经元只与电视图像的统计特征相联,第二个只与电视图像的句法特征相联;第三、第四神经元分别只与红外图像的统计和句法特征相联,而且前四个神经元只和坦克有关,而后四个神经元只与装甲车有关. 在第三层中只有四个神经元,第三层中第一个神经元只与电视图像有关的特征相联,输出只与坦克有关,第二个神经元只与红外图像有关的特征相联,输出也只与坦克有关……学习后的这种网络,对每一个特征都有一定的评价,就是它们在决策中的作用和地位清楚,因此对求解问题的思路提供了十分重要

的依据，在这个例子中，可以得到电视图像的统计特征比句法特征重要，而红外图像中的总体特征则比较重要.

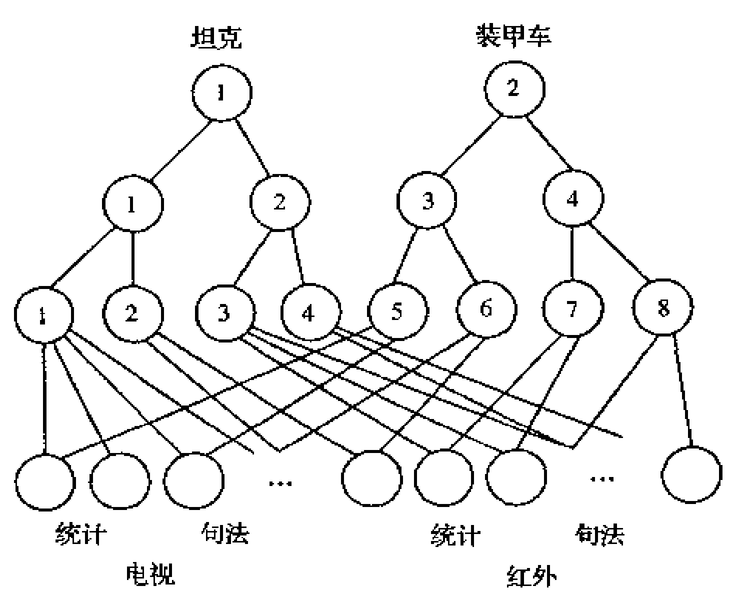


图 8.3 多层分类网络

在模糊系统与神经网络结合的多层网络中，学习的结果是使网络的意义层次清楚，简化后多层决策网络的识别率能超过传统方法，而且又克服了单纯的模糊系统精度问题，使其效果比一般只用单纯的神经网络或者模糊系统要好.

8.2.2 基于模糊推理的神经网络

图 8.4 给出了一种基于模糊推理的神经网络模型. 图中 X 为输入信号，Y 为输出信号，T 为教师信号，白色节点为线性节点，黑色节点为 Sigmoid 型节点. 该系统共分为五层：A 层是输入层，直接将输入值传送至下一层. B 层是输入语言变量层，输出为隶属度函数. C 层为规则层，实现模糊规则前提条件的匹配，执行模糊"与"操作. D 层是输出语言变量层，执行模糊求和操作. E 层为解模糊判决层，产生输出. 可以用有指导的学习方法训练此类模糊神经网络.

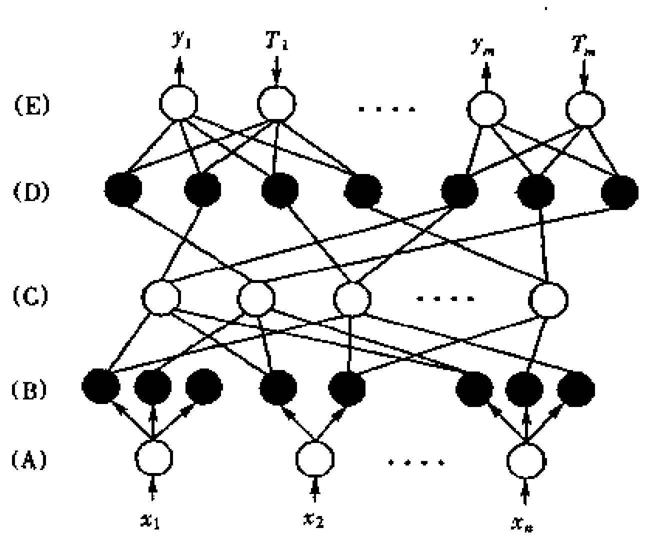


图 8.4 基于模糊推理的神经网络模型

日本的 Hiroyuki Okada 等人提出的用于金融风险评估的模糊神经网络就是一种基于模糊推理的神经网络，其网络模型如图 8.5 所示. 网络也由 Sigmoid 型节点和线性节点构成. 用一个 Sigmoid 型神经元表示 S 型或 Z 型隶属度函数，用两个 Sigmoid 型神经元实现钟形隶属度函数. 它通过领域专家的知识建立模糊模型，确定网络中节点的连接及其权值的初始值. 推理结果用重心法进行非模糊化得到精确的评估结果. 并可以利用应用系统的反馈信息对网络进行训练.

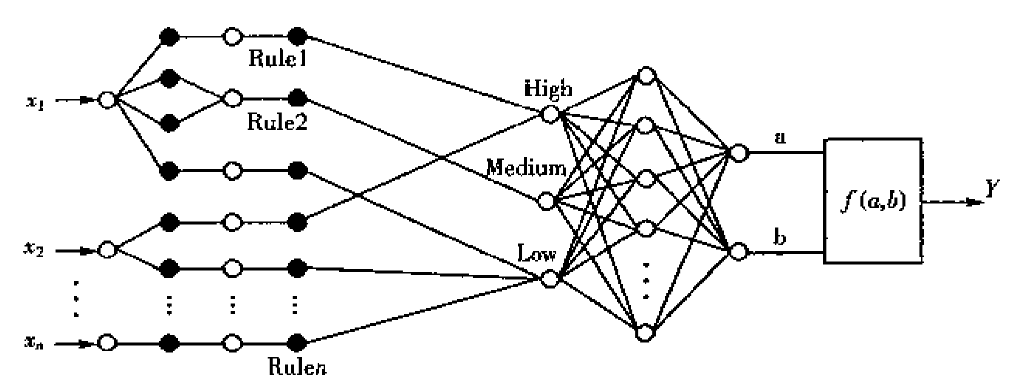


图 8.5　金融风险评估的模糊神经网络模型

●Sidmoid 型神经元　○线性神经元

这种网络的主要特点是用模糊规则确定神经网络结构. 其模糊规则须由领域专家给出，因此能够充分利用领域专家的知识，网络结构简明，模糊规则易于理解，而且网络具有学习能力. 但网络互连结构及权值的确定依赖于领域专家的知识，自适应性差.

8.2.3　基于广义模糊加权型推理法的神经网络

在 6.3.5 节中已对模糊加权型推理法的推理形式进行了改进，提出了广义模糊加权型推理法，现在在此基础上建立一种模糊神经网络.

一、隶属度函数的确定

首先确定输入/输出变量模糊于集的个数. 从理论上讲，在相同论域上，参考模糊于集数越多，越接进真正的最优 FNN，但计算量大，所需内存多，这就需要在最优的精确性和复杂性之间进行权衡. 可以证明，增加模糊子集数使得复杂程度的增加远大于精确度的增加.

另外，通过计算机模拟实验发现，隶属度函数的形状微妙地影响着整个模糊系统. 为了简化计算，很多模糊逻辑控制的隶属度函数曲线取三角形、梯形. 三角形隶属度函数的优点是函数简单，易于计算，但是结果不够平滑，它不是最佳函数，只是一种近似. 实际上，根据模糊统计方法得到的隶属度函数通常都是钟形的，其结果比较平滑，但函数比较复杂，计算量大.

我们试图在精确度和复杂度之间找到一个折衷方案. 用减少模糊于集数来降低系统复杂性，用钟形隶属度函数来提高系统精确度. 为此，规定输入变量的初始隶属度函数如下：

$$\mathrm{big}(x) = \mathrm{Sigmoid}(\alpha_1(x-\beta_1)) = 1/(1-\exp(-\alpha_1(x-\beta_1))) \tag{8.2.1}$$

$$\mathrm{mid}(x) = \exp(-(x-\beta_2)^2/(\alpha_2)^2) \tag{8.2.2}$$

$$\mathrm{small}(x) = 1-\mathrm{Sigmoid}(\alpha_3(x-\beta_3)) = 1-1/(1-\exp(-\alpha_3(x-\beta_3))) \tag{8.2.3}$$

其中 $\alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3 \in \mathbf{R}$.

例如：当 $0 \leqslant x \leqslant 1.0$ 时，可是正规定 x 的初始隶属度函数如图 8.6 所示.

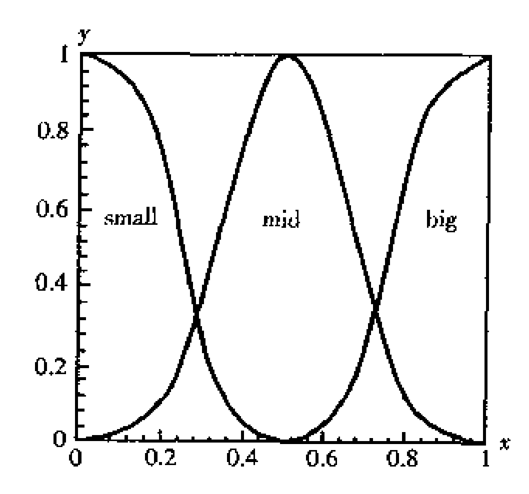


图 8.6　输入 x 的隶属度函数

二、网络结构的确定

在广义模糊加权型推理法的基础之上建立的一种模糊神经网络称为模糊加权神经网络(Weighted-Fuzzy Neural Network，简称 WFNN)，结构如图 8.7 所示.

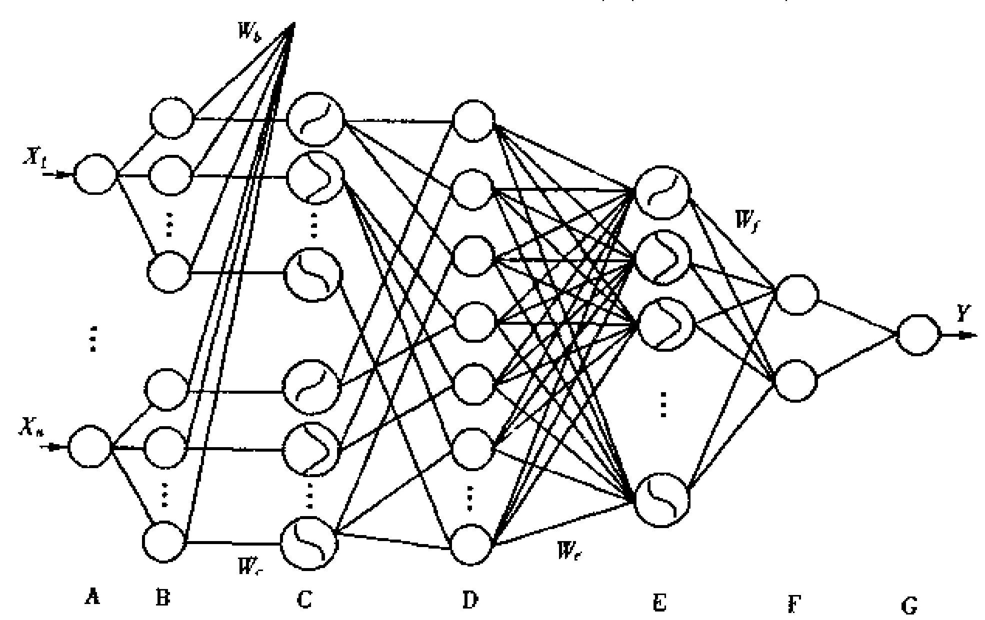


图 8.7　基于模糊加权型推理法的神经网络

图中各层的运算如下所示：

A 层：$O_i = I_i = X_i$

B 层：$O_i = I_i - W_{bi} = X_i - W_{bi}$

C 层：$O_i = \text{Sigmoid}(W_{ci} \times I_i)$ 或

$= \exp(-I_i / W_{ci})^2$ 或

$= 1 - \text{Sigmoid}(W_{ci} \times I_i)$

D 层：$O_i = \prod I_i$

E 层：$O_i = \text{Sigmoid}(\sum (I_i \times W_{ei}))$

F 层：$O_1 = \sum (W_{fi} \times I_i)$；$O_2 = \sum I_i$

G 层：$O = I_1 / I_2$

其中 W_b 为可变阈值，W_c，W_e 和 W_f 为可变权值. 其中，A 层为输入层，B、C 层实现输入量的模糊化，D、E 层实现模糊推理规则，D 层实现模糊规则前提的'与'操作，例如：输入变量的模糊子集定义为{small，mid，big}时，第一个节点表示 if X_1 is small and X_2 is small …then…，第二个节点表示 if X_1 is small and X_2 is mid …then…. E 层完成求和功能，列出规则输出. F、G 层用"重心法"实现非模糊化操作.

由于上述模糊神经网络的节点和权值可以用隶属度函数和模糊规则来解释，因此，模糊规则的抽取和隶属度函数的改进实质上就是寻找合适的 W_b、W_c、W_e 和 W_f. 在确定初始隶属度函数后，用合适的神经网络学习算法，当网络误差小于给定值时，便可以从模糊神经网络的节点和权值参数直接写出模糊规则和改进的隶属度函数，用训练过的网络便可以进行模糊控制、模糊识别了.

三、在天气预报中的应用

下面，通过一个实际的天气预报问题来验证 WFNN 网络模型和算法的有效性. 问题描述如下：由于受各种条件影响，天气是复杂多变的，面且气象科学的许多概念也都是模糊概念，如晴/阴、旱/涝等. 因此采用模糊模型预报天气是非常适宜的. 试建立天气预报模糊神经网络模型，通过对天津市历史汛期降水量的学习，对天津市未来的汛期降水量情况进行预报.

根据天津气象台 1952～1981 年的历史资料，选前一年一月到当年二月的西太平洋海温距平和欧亚地区 500mb 月平均高度距平，分别经主成分分析后，提取的第一个主成分作为两个预报因子，与其对应的降水量值列出如表 8.1 所示.

表 8.1　天津市 1952～1981 年气象数据

年代	海温距平(X_1)	500 mb高度距平(X_2)	降水量(Y)
1952	0.73	−5.28	283
1953	−2.08	5.18	647
1954	−3.53	10.23	731
1955	−3.31	4.21	561
1956	0.53	−2.46	467
1957	2.33	7.32	399
1958	−0.32	−10.81	315
…	…	…	…
1978	0.49	22.63	626
1979	−0.09	−7.32	422
1980	0.04	−6.92	318
1981	0.66	2.37	501

以模糊神经网络 WFNN 为模型，以海温距平(X_1)和 500 mb 高度距平(X_2)为输入，降水量(Y)为输出，输入变量定义了三个模糊子集{small，mid，big}，分别表示小、中、大，隶属度函数如式(8.2.1)、(8.2.2)、(8.2.3)定义. 输出变量定义了五个模糊子集{H_1，H_2，H_3，H_4，H_5}分别表示旱、偏旱、正常、偏涝、涝. 隶属度函数为

$$H_i(x) = \exp(-((x-\beta_4)/\alpha_4)^2) \tag{8.2.4}$$

输入/输出变量隶属度函数曲线如图 8.8 所示.

实验采用遗传算法训练网络，取得了较为满意的结果. 将网络可变阈值 W_b、可变权值 W_c、W_e、W_f 作为基因组成染色体. 每一条染色体对应权值和阈值的一种组合. 群体中第 i 条染色体的适应度为：$S(i)=1/E(i)$，其中 $E(i)=\frac{1}{2}\sum_k (O-T)^2$，$i$ 为染色体个数，k 为学习样本数，T 为教师信号，O 为此条染色体所对应的网络的输出. 初始群体中的染色体数选取 200，以适应度高的染色体被选作父染色体的概率大为原则，每代遗传操作选取 100 对染色体作为父染色体，交叉及变异操作均选取染色体中一半数目的基因，变异概率为 0.1，变异步长为 0.01. 训练中误差曲线如

图 8.9 所示. 计算公式为 $\text{Error}=\frac{1}{2}(O-T)^2$, i 为样本数, O 为网络的实际输出, T 为期望输出. 当网络输出误差小于给定值后进行预报检验.

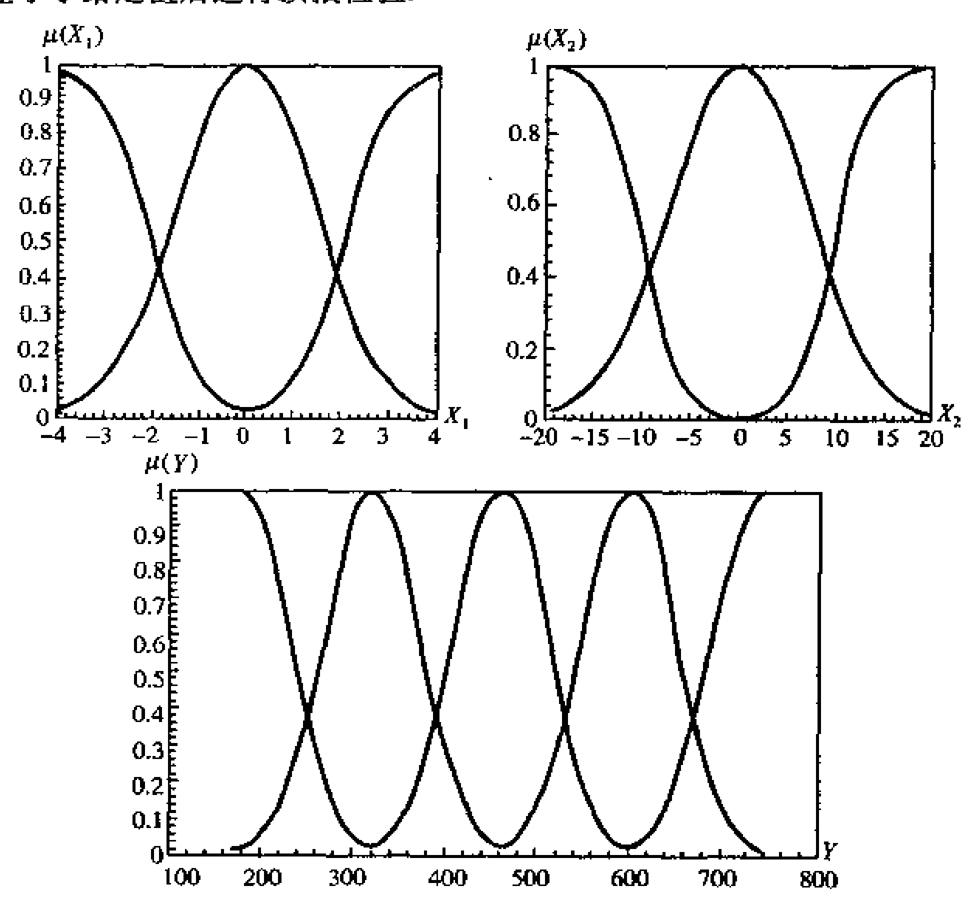


图 8.8　输入/输出变量隶属度函数曲线

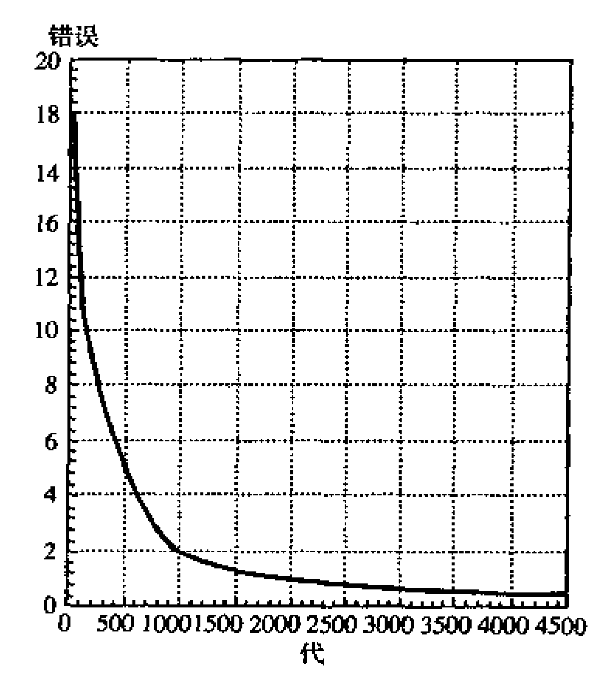


图 8.9　网线误差曲线

应用此模糊神经网络模型，首先对 1952～1973 年进行试报检验，然后对 1973～1981 年进行预报，图 8.10 为输出结果，其中 1973～1981 为预报值，网络误差在允许范围之内，可见效果相当理想．说明所建立的模糊神经网络模型是有效的．

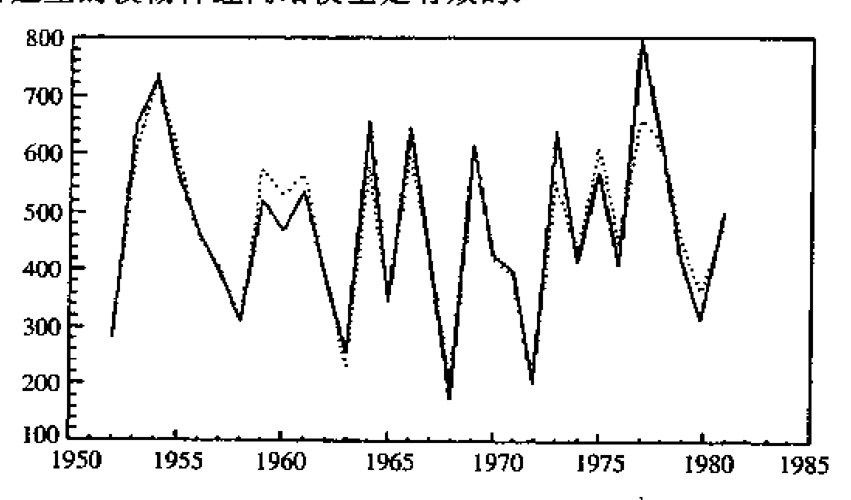


图 8.10　天气预报输出结果

——实测值　……预测值

各网络权值在训练前后的变化如表 8.2(a～d)所示．

表 8.2　各网络权值在训练前后的变化

(a)W_b 的变化

	W_{b0}	W_{b1}	W_{b2}	W_{b3}	W_{b4}	W_{b5}
学习前	2	0	−2	10	0	−10
500 代后	1.85	0.23	−1.7	9.95	0.08	−9.87
1000 代后	1.60	0.73	−1.32	9.80	−0.06	−9.61
4000 代后	0.72	0.52	−1.27	9.30	1.57	−9.51
5000 代后	0.79	0.55	−0.45	9.25	1.75	−8.86

(b)W_c 的变化

	W_{c0}	W_{c1}	W_{c2}	W_{c3}	W_{c4}	W_{c5}
学习前	2	2	2	0.5	10	0.5
500 代后	1.91	1.95	2.07	0.26	9.98	0.66
1000 代后	1.65	1.71	1.77	·0.1	9.82	0.42
4000 代后	1.97	0.28	1.83	1.23	9.45	0.84
5000 代后	2.40	0.29	2.28	1.25	9.27	1.42

(c)W_f 的变化

	W_{f0}	W_{f1}	W_{f2}	W_{f3}	W_{f4}	
学习前	180	320	460	600	740	
500 代后	147	292	459	616	759	
1000 代后	111	256	435	592	744	
4000 代后	83	334	477	541	839	
5000 代后	51	301	495	478	878	

(d)W_e的变化

W_e	代数(epochs)				W_e	代数(epochs)				W_e	代数(epochs)			
ij	500	1000	4000	5000	ij	500	1000	4000	5000	ij	500	1000	4000	5000
00	1.0	0.66	0.48	0.50	30	1.24	0.92	0.84	1.18	60	1.45	1.70	1.60	1.51
01	0.77	0.67	0.43	0.37	31	1.05	1.16	0.57	0.49	61	1.32	1.29	0.86	0.87
02	0.81	0.95	1.67	1.49	32	0.88	0.99	1.99	2.20	62	0.71	0.50	0.08	0.33
03	0.89	1.02	0.20	−0.02	33	0.80	0.86	0.10	0.66	63	0.63	0.39	0.70	0.92
04	1.06	0.83	0.79	0.75	34	0.68	0.47	0.01	0.10	64	0.56	0.20	−0.03	−0.21
10	0.74	0.63	1.27	1.40	40	0.85	0.78	1.55	1.52	70	1.06	0.88	0.55	0.17
11	0.73	0.59	0.32	0.64	41	0.86	0.82	0.56	0.66	71	0.96	0.86	1.08	1.18
12	0.93	1.16	1.63	1.36	42	0.93	0.88	0.80	0.90	72	0.88	0.79	0.11	1.19
13	1.11	1.19	1.36	1.69	43	0.99	0.85	0.22	0.50	73	0.89	0.80	1.36	1.60
14	1.21	0.80	0.08	−0.11	44	0.97	0.70	1.19	1.32	74	0.92	0.64	0.69	0.85
20	0.63	0.27	0.59	0.15	50	0.79	0.58	1.48	1.88	80	0.96	0.72	0.30	0.65
21	0.68	0.34	−0.35	−0.66	51	0.85	0.78	0.02	0.47	81	0.86	0.62	0.47	0.10
22	0.83	0.61	0.19	−0.49	52	0.94	0.77	0.31	0.02	82	0.85	0.63	−0.19	−0.34
23	1.28	1.28	0.53	0.85	53	1.03	0.84	0.08	0.12	83	0.92	0.91	0.91	0.73
24	1.39	1.62	2.24	2.09	54	1.05	0.86	1.54	1.48	84	1.02	0.84	1.05	1.27

对于表 8.2(a～d)，W_b的变化可解释为 X_i的隶属度函数中心的变化，W_c的变化可解释为 X_i的隶属度函数幅度的变化，W_f的变化可解释为 Y 的隶属度函数中心的变化. 而通过对 W_e的分析，可得出模糊规则相对重要性的变化，如：

rule 1：if X_1 is small and X_2 is small then Y is H_1…1.0，0.66，0.48，0.50

rule 45：if $X_1$is big and X_2 is big tben Y is H_5…1.02，0.84，1.05，1.27

可见，rule 1 的重要性下降了，而 rule 45 的重要性提高了. 按此方法，可提取出重要性提高的 9 条规则作为实际应用的参考. 规则如下所示：

rule 1：if X_1 is small and X_2 is small then Y is H_3 with 1.49

rule 2：if $X_1$is small and X_2 is mid then Y isH_4 with 1.69

rule 3：if $X_1$is small and X_2 is bigs then Y is H_5 with 2.09

rule 4：if X_1 is mid and X_2 is small then Y is H_3 with 2.20

rule 5：ifX_1 is mid and X_2 is mid then Y is H_1 with 1.52

rule 6：if X_1 is mid and X_2 is big then Y is H_1 with 1.88

rule 7：if X_1 is big and X_2 is small then Y is H_1 with 1.51

rule 8：if $X_1$is big and X_2 is mid then Y is H_4 with 1.60

rule 9：if X_1 is big and X_2 is big then Y is H_5 with 1.27

在训练 1000 代后的输入/输出变量隶属度函数变化如图 8.11 所示.

四、在味觉信号识别中的应用

在味觉信号识别中用到的网络结构如图 8.12 所示. 图中 Net-A 部分用于混合味的学习训练，Net-B 部分用于单一味的学习训练. 可见 Net-A 即为 WFNN 的结构，而 Net-B 为简化的 WFNN 结构，取消了进行非模糊化操作的 F 层和 G 层，其 E 层的输出与各单一味相对应. 网络输入为味觉信号的特征，Net-A 输出为 y_{mix}，标识各混合味. Net-B 输出为 $y_i(i=1,2,\cdots,m)$，对应于各单一味，当学习样本为五种基本味时，$m=5$. 网络最终通过解模糊判决输出味觉评价. Net-A 与 Net-B 各层的运算与 WFNN 相同.

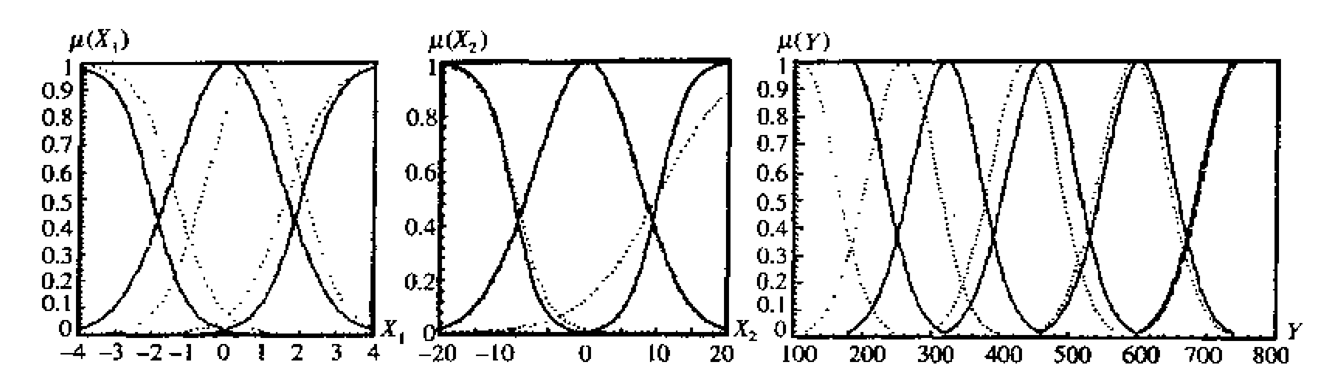


图 8.11　输入/输出变量隶属度函数曲线的变化

图 8.12　用于味觉信号识别的模糊神经网络模型

根据前面的实验，对采样的味觉信号数据进行数据压缩后，获得的数据样本如表 8.3 及表 8.4 所示. 其中表 8.3 为单一味的学习样本，将其输入到 Net-B 进行训练. 表 8.4 为酸甜混合味的学习样本，将其输入到 Net-A 进行训练. 这是经一次采样获得的数据样本，如果进行多次采样，就可以获得多个样本. 样本数目越多，网络学习后的精度越好.

表 8.3　单一味学习样本

样本代号	味觉评价	输入				输出				
		X_1	X_2	X_3	X_4	y_1	y_2	y_3	y_4	y_5
Taste 1	酸味	0.197009	0.201018	0.489089	0.615916	1	0	0	0	0
Taste 2	甜味	0.256033	0.253065	0.961945	0.811039	0	1	0	0	0
Taste 3	咸味	0.183008	0.186057	0.917072	0.322078	0	0	1	0	0
Taste 4	苦味	0.256907	0.256001	0.686084	0.368040	0	0	0	1	0
Taste 5	香味	0.145057	0.151054	0.737945	0.565909	0	0	0	0	1

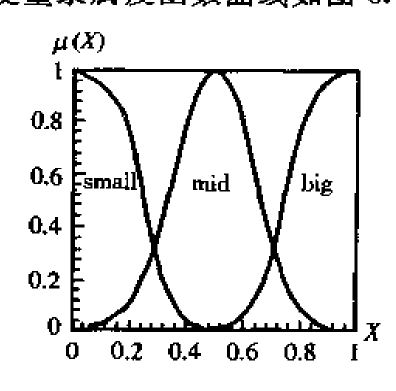
表 8.4　混合味学习样本

样本代号	味觉评价	输入				输出
		X_1	X_2	X_3	X_4	y_{mix}
Mix 1	纯酸味	0.197009	0.201018	0.489089	0.615916	1.0
Mix 2	纯酸味	0.212433	0.215684	0.518368	0.762794	1.0
Mix 3	酸甜适中	0.263253	0.273787	0.978864	0.846897	0.5
Mix 4	甜酸味	0.283425	0.302352	0.992352	0.892352	0.25
Mix 5	酸甜味	0.245363	0.254574	0.789557	0.868546	0.75
Mix 6	酸甜适中	0.269769	0.278976	0.971353	0.841235	0.5
Mix 7	纯甜味	0.256033	0.253065	0.961945	0.811039	0.0
Mix 8	纯甜味	0.268934	0.271243	0.977980	0.855759	0.0

根据学习样本我们确定图 8.12 中的输入为 $X_1 \sim X_4$，将输入变量定义了三个模糊子集{small，mid，big}，分别表示小、中、大，隶属度函数如式(8.6.1)、(8.6.2)、(8.6.3)定义. 对于Net-A，输出变量定义了五个模糊子集$\{T_1, T_2, T_3, T_4, T_5\}$，分别表示纯酸味、酸甜味、酸甜适中、甜酸味、纯甜味，隶属度函数为

$$T_i(x) = \exp(-((x-\beta_i)/\alpha_i)^2) \tag{8.2.5}$$

输入/输出变量隶属度函数曲线如图 8.13 所示.



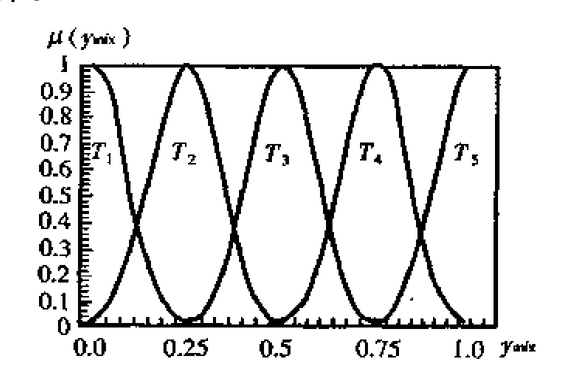


图 8.13　输入/输出变量隶属度函数曲线

仍采用遗传算法训练网络，将网络可变阈值及可变权值作为基因组成染色体. 群体中第 i 条染色体的适应度为：$S(i)=1/E(i)$，其中 $E(i)=\frac{1}{2}\sum_k (O-T)^2$，$i$ 为染色体个数，k 为学习样本数，T 为教师信号，O 为此条染色体所对应的网络的输出. 初始群体中的染色体数选取500，每代遗传操作选取 250 对染色体作为父染色体，交叉及变异操作选取染色体中一半数目的基因，变异概率为 0.1，变异步长为 0.01. 当网络输出误差小于给定值时，学习过程即告结束，此时将最终调整好的权值存入文件以待识别时使用.

Net-A 对混合味的训练误差曲线如图 8.14 所示. 其中变异率分别取 0.01,0.05,0.1,0.2,0.9. 在训练 4000 代后,网络误差小于 0.01,各网络权值和阈值的变化如表 8.5 所示.

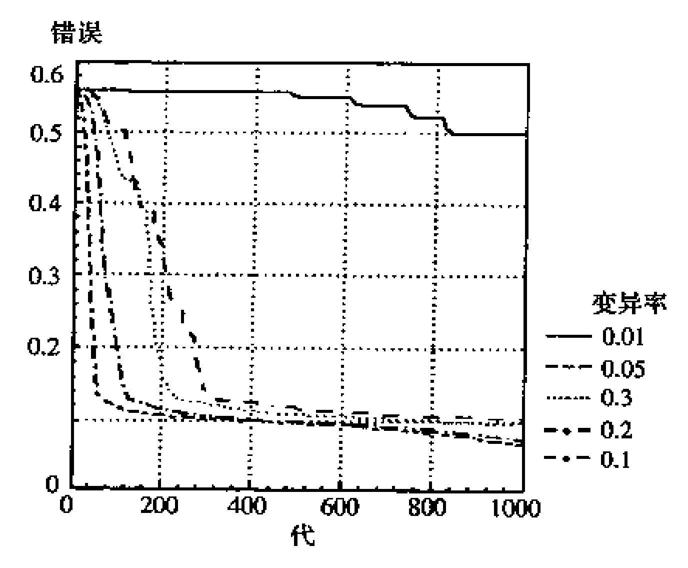


图 8.14　遗传算法训练模糊神经网络的误差曲线

表 8.5　各网络权值在训练前后的变化

(a)W_b的变化

	W_{b0}	W_{b1}	W_{b2}	W_{b3}	W_{b4}	W_{b5}	W_{b6}	W_{b7}	W_{b8}	W_{b9}	W_{b10}	W_{b11}
学习前	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
500 代后	0.26	0.47	0.76	0.24	0.25	0.72	0.17	0.64	0.81	0.26	0.47	0.73
2000 代后	0.28	0.52	0.79	0.23	0.31	0.75	0.27	0.67	0.81	0.23	0.35	0.70
4000 代后	0.29	0.53	0.70	0.24	0.32	0.73	0.28	0.63	0.82	0.24	0.45	0.73

(b)W_c的变化

	W_{c0}	W_{c1}	W_{c2}	W_{c3}	W_{c4}	W_{c5}	W_{c6}	W_{c7}	W_{c8}	W_{c9}	W_{c10}	W_{c11}
学习前	20	0.2	20	20	0.2	20	20	0.2	20	20	0.2	20
500 代后	20.06	0.26	20.02	20.00	0.41	20.09	19.99	0.17	20.01	20.05	0.16	19.93
32000 代后	19.98	0.29	20.12	20.21	0.21	20.07	19.99	0.18	20.03	20.07	0.13	19.90
4000 代后	19.94	0.24	20.30	20.34	0.18	20.10	19.98	0.19	20.07	19.99	0.10	19.91

(c)W_f的变化

	W_{f0}	W_{f1}	W_{f2}	W_{f3}	W_{f4}
学习前	0.0	0.25	0.5	0.75	1.0
500 代后	−0.10	0.25	0.74	1.07	1.23
2000 代后	−0.12	0.15	0.62	0.88	1.18
4000 代后	−0.11	0.14	0.58	0.84	1.13

(d)W_e的变化

	W_{e00}	W_{e01}	W_{e02}	W_{e03}	W_{e04}
学习前	1.0	1.0	1.0	1.0	1.0
500 代后	0.98	0.96	0.97	1.04	0.99
2000 代后	0.97	0.99	0.97	1.02	0.90
4000 代后	0.91	0.99	0.93	1.07	0.88

同样,可以将 W_b的变化解释为 X_i的隶属度函数中心的变化,W_c的变化解释为 X_i的隶属度函数幅度的变化,W_f的变化可解释为 y_{mix}的隶属度函数中心的变化. 而通过对 W_e的分析,可

得出模糊规则相对重要性的变化，如：

rule 1：if X_1 is small and X_2 is small and X_3 is small and X_4 is small then

y_{mix} is T_1…1.0，0.98，0.97，0.91

rule 4：if X_1 is small and X_2 is small and X_3 is small and X_4 is small then

y_{mix} is T_4…1.0，1.04，1.02，1.07

可见，rule 1 的重要性下降了，而 rule 4 的重要性提高了. 这样我们就可以保留规则 4 而舍弃规则 1. 按此方法，可提取出重要性提高的若干条规则作为实际应用的参考. 训练 4000 代后的输入输出变量隶属度函数变化如图 8.15 所示.

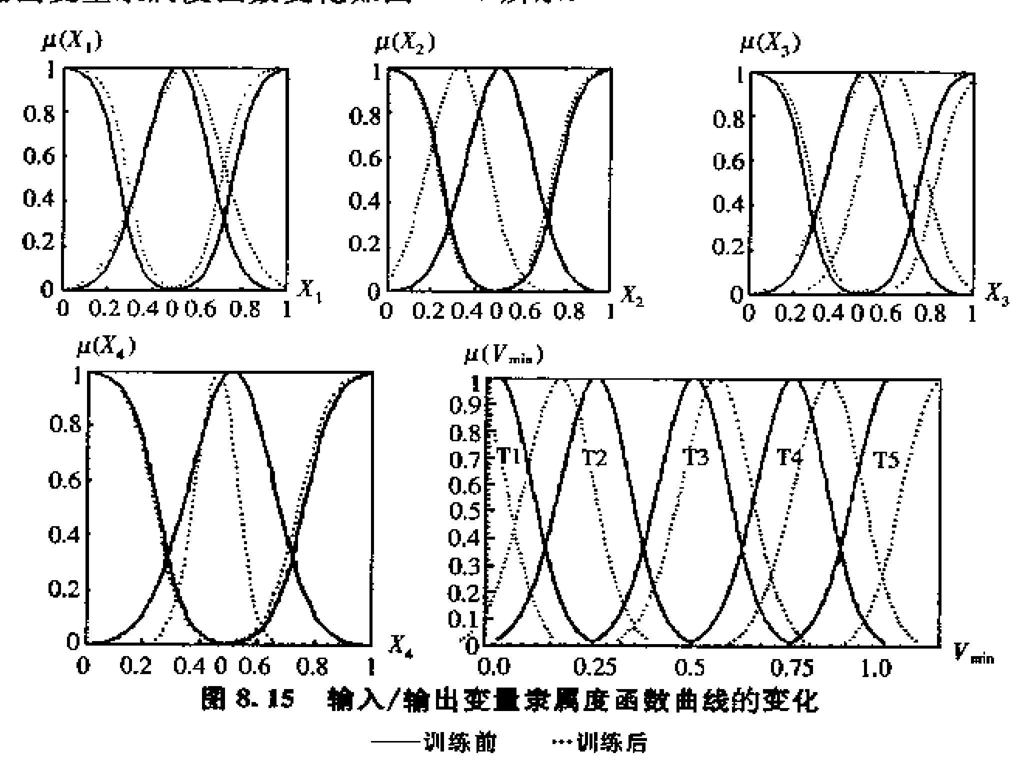


图 8.15　输入/输出变量隶属度函数曲线的变化

——训练前　…训练后

网络经训练后，我们就可以对已知和未知的味觉信号进行识别了. 对味觉信号的识别过程如图 8.16 所示.

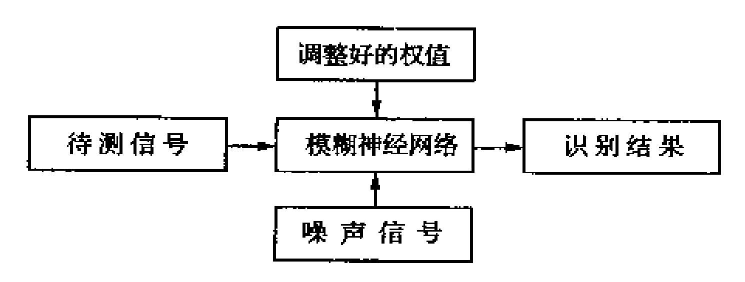


图 8.16　味觉信号的识别过程

将味觉信号的特征输入到网络中，根据 Net-A 和 Net-B 的输出进行判决，给出味觉评价. 判决方法是：如果已知待测信号为某种单一味时，可以根据 Net-B 的输出中哪一个 y_i 为逻辑'1'给出味觉评价. 如果已知待测信号为混合味时(在本文中为酸甜混合味)，则可以根据 Net-A 的输出 y_{mix} 及其隶属度函数给出味觉评价. 如果待测信号为未知信号时，则按如下方法判决：如果 Net-B 的输出 y_i 中有且仅有一个为'1'，则判断此味觉为单一味，并根据哪一个 y_i 为'1'给出味觉评价；否则判断此味觉为混合味，并根据 y_{mix} 的输出及隶属度函数给出味觉评价.

逻辑‘1’须设定某个实数区间，如[0.8,1]，只要输出在此区间就认为是‘1’.

实验结果表明：对于训练过的样本识别率接近100%. 当待测信号为五种基本味觉时，其输出结果及味觉评价如表8.6所示. 当待测信号为混合味时，其输出结果及味觉评价如表8.7所示.

表 8.6　基本味信号识别结果

样本代号	实际输出					味觉评价
	y_1	y_2	y_3	y_4	y_5	
Taste 1	0.952079	0.022198	0.021339	0.020423	0.01000	酸味
Taste 2	0.028328	0.981666	0.010000	0.010000	0.01000	甜味
Taste 3	0.010000	0.010000	0.99000	0.014723	0.010000	咸味
Taste 4	0.010678	0.01000	0.01000	0.990000	0.010000	苦味
Taste 5	0.010000	0.010000	0.01000	0.038766	0.987471	香味

表 8.7　混合味信号识别结果

样本代号	实际输出 y'_{mix}	期望输出 y_{mix}	味觉评价
mix 1	0.96	1.0	纯酸味
mix 2	1.01	1.0	纯酸味
mix 3	0.41	0.5	酸甜适中
mix 4	0.27	0.25	甜酸味
mix 5	0.73	0.75	酸甜味
mix 6	0.49	0.5	酸甜适中
mix 7	0.04	0.0	纯甜味
mix 8	0.09	0.0	纯甜味

上面的结果是使用标准味觉信号，即学习过程中所使用的输入信号，但是在实际测量过程中的输入信号往往偏离标准，为考证模糊神经网络对味觉信号的识别精度，可在标准信号上附加随机噪声信号作为输入信号. 设 A 为标准信号，B/A 为信噪比，C 为[−1,1]间的随机数，那么附加噪声信号后的输入信号值 A' 为：

$$A' = A + C \times (B/A) \times A$$

图8.17给出了在 B/A 为0.1时，用Net-A对mix1～mix8识别200次的结果误差分布图，误差计算公式为：

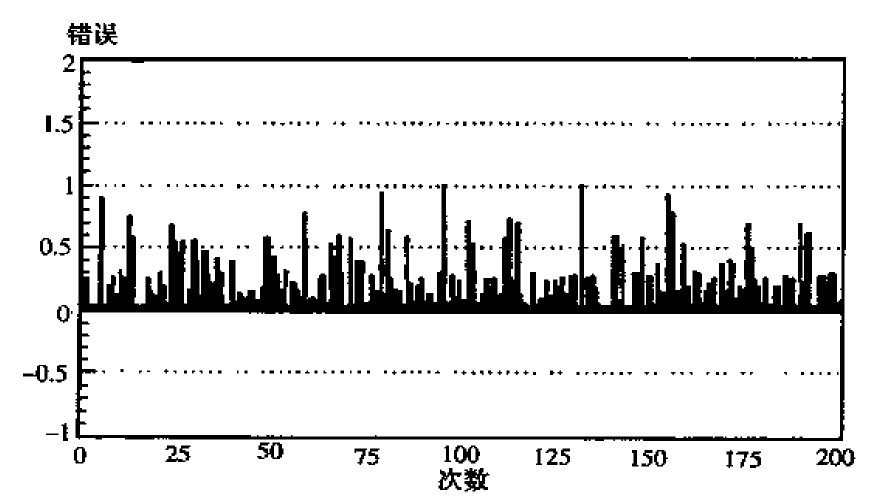


图 8.17　B/A=0.1 时的网络误差分布

$$\text{Error} = \frac{1}{2}\sum_i (O - T)^2$$

式中 i 为样本数，O 为网络的实际输出，T 为期望输出. 其中 Error 小于 0.5 为 165 次，达 82.5%，可见网络精度是很好的.

信噪比 B/A 对网络误差的影响如图 8.18 所示，其中 B/A 取值为 0.0～1.0，间隔为 0.01. 对每一个 B/A，用网络识别 1000 次，并计算网络输出的平均误差 Error. 可见当信噪比小于 0.2 时网络精度仍很高.

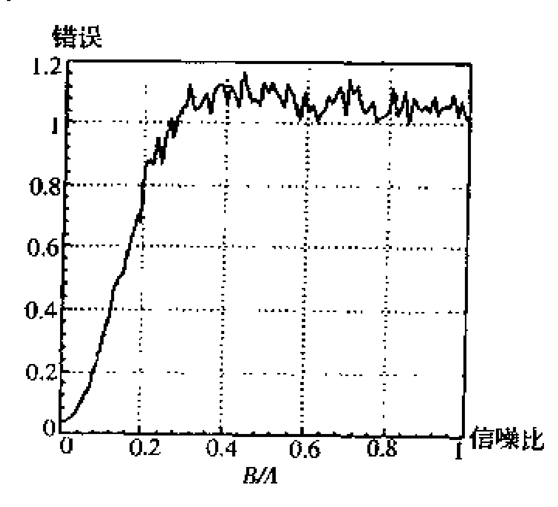


图 8.18 网络输出误差随 B/A 变化曲线

下面，用如图 8.19 所示的三层前向神经网络对单一味觉信号进行识别，输入 X_i 为味觉信号特征，输出 y_i 对应各单一味觉信号. 分别用 BP 算法和遗传算法对其进行训练，旨在与模糊神经网络模型 WFNN 进行对比. 实验表明：GA 和 BP 算法相比，虽然 GA 训练时间较长，大约是 BP 算法的 1.5 倍以上，但 GA 在全局搜索方面优于 BP 算法，GA 和 BP 算法相比陷入局部极小的概率低. 从图 8.20 可以看出，用 GA 进行训练模糊神经网络模型 WFNN 与三层前向神经网络的误差收敛速度相当，WFNN 较好一些. 可见模糊神经网络模型 WFNN 较简单的三层前向神经网络，不仅在网络结构上清晰、易于理解，而且在功能上也是实用的、有效的.

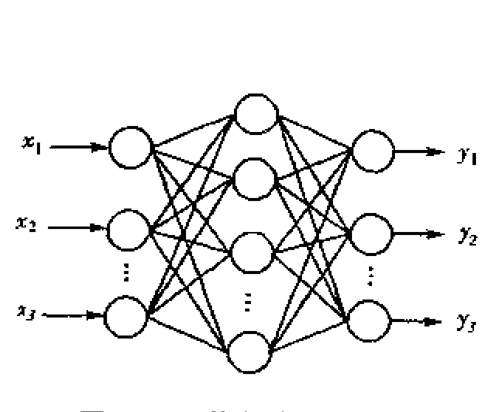


图 8.19 前向神经网络图

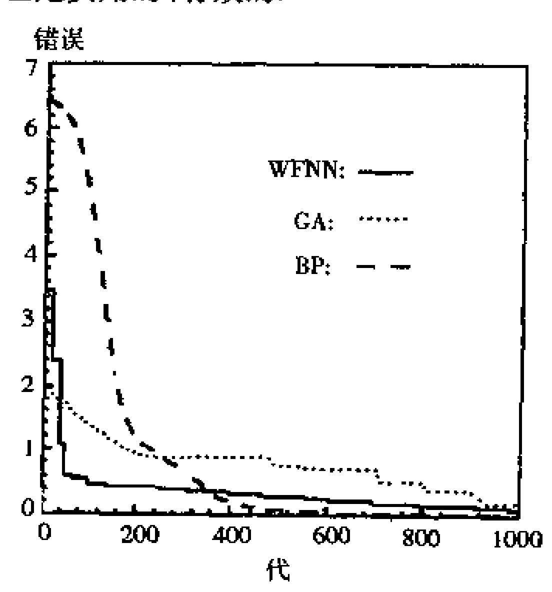


图 8.20 几种网络及算法的比较

8.2.4 模糊神经网络和前馈型网络的组合式网络模型

模糊神经网络和前馈型网络的组合式网络模型不需要领域专家的知识进行指导，而是通过样本的竞争分类产生模糊规则. 每类样本对应一条模糊规则，每条规则的后件部分为一个对本类样本进行学习训练的前馈型神经网络，可用遗传算法进行学习训练. 该模糊神经网络能够较好地处理拒识样本的问题，并且在时间序列分析的应用中取得了较好的效果.

一、网络模型

该网络模型由四层不同功能的节点构成，见图 8.21，A 层为模糊化层，B 层为规则层，C 层为后件神经网络层，D 层实现非模糊化. 网络包含 C 条规则，每条模糊规则具有以下形式：

$$\text{if } X_1 \text{ is } L_{1j} \text{ and } X_2 \text{ is } L_{2j} \text{ and } \cdots \text{and } X_N \text{ is } L_{Nj} \text{ then } Y = \text{net}_j(X_1, X_2, \cdots, X_N)$$

其中 L_{ij} 为对应输入变量 X_i 的第 j 个语言变量，net_j 为第 j 条规则的后件神经网络的映射函数，$i=1,2,\cdots,N$；$j=1,2,\cdots,C$.

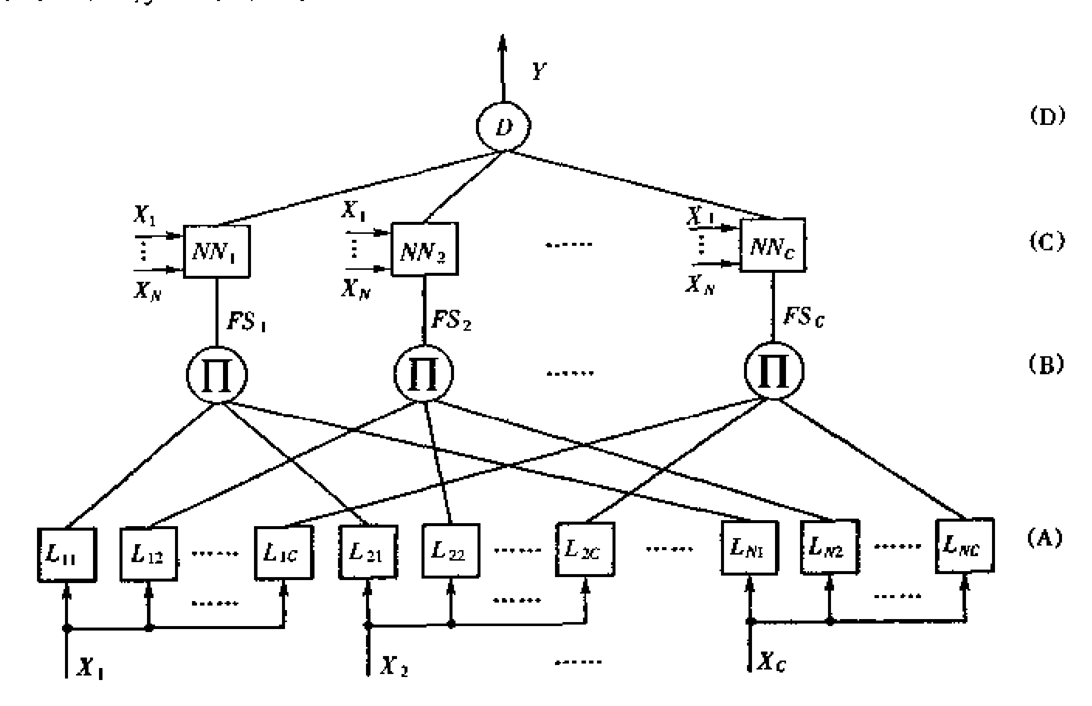


图 8.21 模糊神经网络和前馈型网络的组合式网络模型

A 层为语言变量隶属度函数层. 每个输入 X_i 划分为 C 个语言变量，每一个节点对应一个语言变量，节点的输出为隶属度函数值. 节点的传输函数为：

$$O_{ij}^1 = \mu_{Lij}(X_i) \tag{8.2.6}$$

B 层为取最小层. 该层节点对 A 层节点给出的隶属度值进行取最小或代数乘操作，得到各条规则的点火强度(Firing Strength). 节点的传输函数为：

$$O_i^2 = FS_i = \min_{1 \leqslant i \leqslant C}\{O_{1i}^1, O_{2i}^1, \cdots, O_{Ni}^1\} \text{ 或者 } O_i^2 = FS_i = \prod_{j=1}^{N} O_{ji}^1 \tag{8.2.7}$$

其中，FS_i 为第 i 条规则的点火强度，$i=1,2,\cdots,C$.

C 层为后件神经网络层. 每条规则的后件部分对应于一个有两个隐层的前馈型神经网络. 输入向量为 $\boldsymbol{X}=(X_1, X_2, \cdots, X_N)$，并对本类样本进行学习训练. 后件神经网络的输出与本规则的点火强度相乘作为该层的输出. 节点的传输函数为：

$$O_i^3 = O_i^2 \mathrm{net}_i(\boldsymbol{X}) = FS_i \mathrm{net}_i(\boldsymbol{X}) \tag{8.2.8}$$

其中，net_i为第 i 条规则的后件神经网络的映射函数，$i=1,2,\cdots,C$.

D层为非模糊化层. 该层对各个后件神经网络的输出结果用重心法进行非模糊化，产生模糊神经网络的输出结果. 节点的传输函数为：

$$Y = O^4 = \frac{\sum_{i=1}^{C} O_i^3}{\sum_{i=1}^{C} FS_i} = \frac{\sum_{i=1}^{C} [\mathrm{net}_i(X) FS_i]}{\sum_{i=1}^{C} FS_i} \tag{8.2.9}$$

二、样本竞争分类

为了确定模糊规则前件中语言变量的隶属度函数，首先对学习样本进行竞争分类. 假设共有 M 个输入样本 $\boldsymbol{T}_1,\boldsymbol{T}_2,\cdots,\boldsymbol{T}_M$，其输入向量为 $\boldsymbol{T}_k=(T_{k1},T_{k2},\cdots,T_{kN})$，$k=1,2,\cdots,M$. 用如下方法对输入样本进行竞争分类：

①在输入样本空间中投入 C 个“种子”，$\boldsymbol{S}_i=(S_{i1},S_{i2},\cdots,S_{iN})$，$i=1,2,\cdots,C$.

②对每个输入样本和每个种子计算 $D_{ik}=\sum_{i=1}^{N}|S_{ij}-T_{kj}|$，具有最小 D_{ik} 值的种子 $\boldsymbol{S}_i$ 就是对输入样本 $\boldsymbol{T}_k$ 竞争的胜者，即输入样本 T_k 是第 i 类的成员. 每个输入样本只有一个竞争胜者，即每个输入样本只属于 C 类中的某一类.

③对所有的样本竞争完毕后，对每个种子根据竞争获胜的情况按下式进行修正.

$$\boldsymbol{S}_i = \boldsymbol{S}_i + \alpha \sum_{k=1}^{M} (\boldsymbol{T}_k - \boldsymbol{S}_i) \tag{8.2.10}$$

其中 α 为学习率，取较小的正数 $\boldsymbol{S}_i$ 为 $\boldsymbol{T}_k$ 竞争胜者.

④如果竞争次数已到则结束，否则转②.

在竞争分类过程中对种子的修正，使得种子在竞争到的样本的共同作用下向得到的样本移动. 经过多次竞争和修正，种子最终稳定在竞争所得到的样本中心. 因为每个输入样本仅有一个获胜者，所以每个样本仅属于一个类，这样经过竞争，学习样本分为 C 类.

三、输入变量隶属度函数的确定

对于 C 类学习样本，每类对应一条模糊规则. 模糊规则前件中语言变量的隶属度函数采用图 8.22 所示的梯形函数，其形状由参数 a,b,c,d 来确定. 语言变量 L_{ij} 的参数由下式确定.

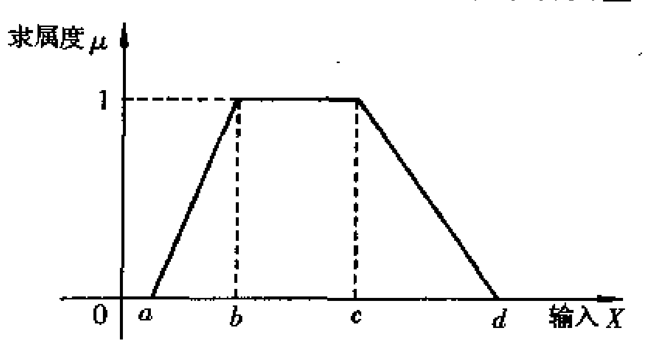


图 8.22　梯形函数

$$b_{ij} = \min\{T_{ki} \mid T_k \in j\}$$
$$c_{ij} = \max\{T_{ki} \mid T_k \in j\}$$

$$a_{ij}=\begin{cases}-\infty, & b_{ij}=\min\limits_{1\leqslant k\leqslant M}\{T_{ki}\}\\ \max\{T_{ki}\mid T_{ki}<b_{ij}\}, & \text{否则}\end{cases}$$

$$d_{ij}\begin{cases}+\infty, & d_{ij}=\max\limits_{1\leqslant k\leqslant M}\{T_{ki}\}\\ \min\{T_{ki}\mid T_{ki}>C_{ij}\}, & \text{否则}\end{cases}$$

输入变量 X 的隶属度计算如下：

$$\mu(x)=\begin{cases}0, & x\geqslant d \text{ 或 } x\leqslant a\\ \dfrac{x-a}{b-a}, & b>x>a\\ 1, & c\geqslant x\geqslant b\\ \dfrac{d-x}{d-c}, & d>x>c\end{cases} \tag{8.2.11}$$

四、模糊规则后件子网的训练

后件子网采用遗传算法进行训练. 每个子网学习属于本类的学习样本，训练一定次数或误差满足要求后结束训练.

五、拒识样本的处理

经过竞争，学习样本分为 C 类，N 维输入向量的每一维对应 C 个语言变量，共有 C^N 种语言变量的组合，即有 C^N 条规则. 当 N 较大时，C 条规则仅是 C^N 种组合的一小部分. 实际上往往会发生拒识的情况，即所有规则的点火强度均为零，所有的规则都没被激活. 对于拒识样本 $I=(I_1,I_2,\cdots,I_N)$，计算 I 与每个种子的距离 $D_i^*=\sum\limits_{j=1}^{N}|S_{ij}-I_j|$，$i=1,\cdots,C$，以 $1/D_i^*$ 作为各规则的点火强度，模糊神经网络的输出结果为：

$$Y=O^4=\frac{\sum\limits_{i=1}^{C}[\text{net}_i(I)/D_i^*]}{\sum\limits_{i=1}^{C}I/D_i^*} \tag{8.2.12}$$

这样，用拒识样本与各条规则对应的种子(即类的中心)的距离确定规则的激活程度，距离越大，样本越偏离这条规则，规则的激活程度越小. 综合所有规则的结果，用重心法进行非模糊化即可得到一个能代表全体规则输出结果的值，从而解决了样本拒识的问题.

六、在时间序列分析中的应用

现将模糊神经网络和前馈型网络的组合式网络模型用于时间序列分析上，并和另外两种方法相比较以验证其有效性.

这是一个气温预测的实验. 数据来源是武汉市 1978 年 1 月至 1987 年 12 月的月平均气温数据，共 120 个. 实验采用图 8.21 所示的模糊神经网络模型，其中包含 4 条规则，后件子网有 12 个输入，一个输出，有两个隐层，隐层单元数均为 30. 用前 12 个数据为输入，进行向前一步预测，得到下一个数据. 这样 120 个数据可以产生 108 个输入输出模式，以其中前 90 个模式为学习样本，其他的作为测试样本. 每个子网用遗传算法训练 5000 代. 模糊神经网络输出结果与样本输出的比较如图 8.23 所示.

在时间序列分析和系统辨识中，传统上使用的是 ARMA(Auto Regressive Moving Average)自动回归滑动平均模型，但由于 ARMA 模型的建立比较复杂，实际应用中常用 AR(Auto

Regressive)模型. 本实验中采用 AR(12)模型, 用 12 个数据作为输入, 进行向前一步测试, 得到下一个数据. AR(12)的参数确定为:

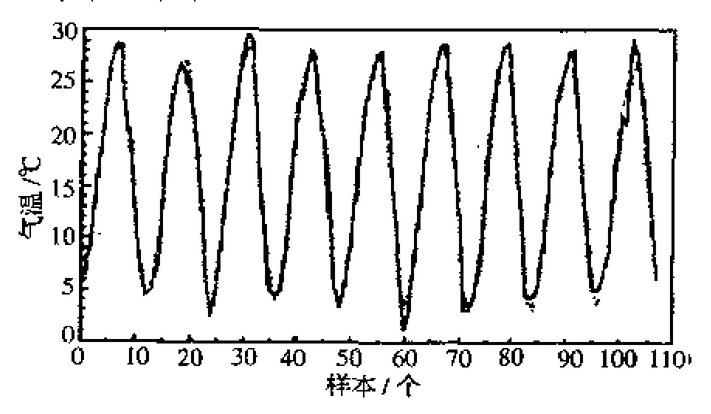


图 8.23　模糊神经网络输出结果与样本输出的比较

——模糊神经网络　——样本

$$X_t = 0.728X_{t-1} - 0.246X_{t-2} - 0.080X_{t-3} - 0.209X_{t-4} - 0.203X_{t-5} + 0.018X_{t-6} - 0.375X_{t-7} + 0.052X_{t-8} + 0.007X_{t-9} - 0.114X_{t-10} + 0.043X_{t-11} - 0.073X_{t-12} \quad (8.2.13)$$

第三种方法是前向神经网络方法, 即将时间序列预测中的输入输出模式作为网络的学习样本来训练网络, 利用网络的学习能力从样本中获取知识, 从而做出预测. 前向神经网络方法的主要问题是当样本较多时网络的训练难以收敛. 实验采用一个具有 12 个输入节点, 一个输出节点, 40 个单隐层节点的前向网络, 对 90 个样本进行学习训练, 其他样本用来进行测试.

三种方法的测试结果对比如图 8.24 所示, 可以看出, 模糊神经网络用于时间序列分析上取得了较好的效果.

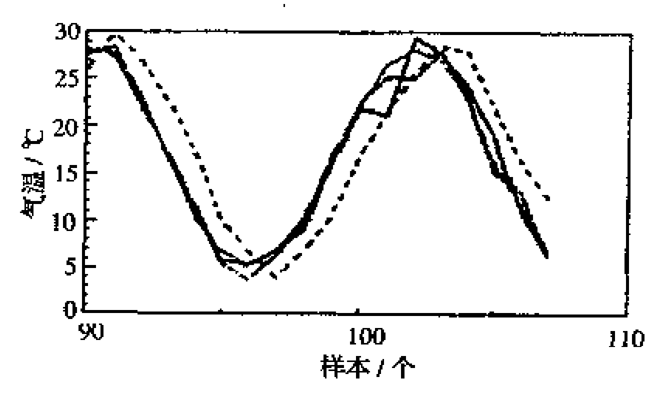


图 8.24　三种方法用于预测气温的比较

——样本　—·—模糊神经网络　···前向神经网络　—AR

§8.3　神经模糊系统

8.3.1　基于神经网络的自适应模糊控制器

自适应模糊控制在模糊控制学科体系中占有特别重要的地位. 自从 1979 年英国的 Mamdani E H 和他的学生 Procyk T J 首先提出自适应模糊控制以来, 自适应模糊控制领域已取得了丰硕的成果, 尤其是引入神经网络学习方法以后, 基于神经网络的自适应模糊控制新方法层出不穷, 成为引人瞩目的研究热点.

汽车速度控制器是一个基于神经网络的自适应模糊控制器的例子. 现有汽车速度和道路的不平两个输入，以及一个输出，即油门位置. 汽车速度由转速表转换而来；道路不平度是由路而等级得到.

首先，如图 8.25 所示，量化并选择两个输入与一个输出的模糊子集.

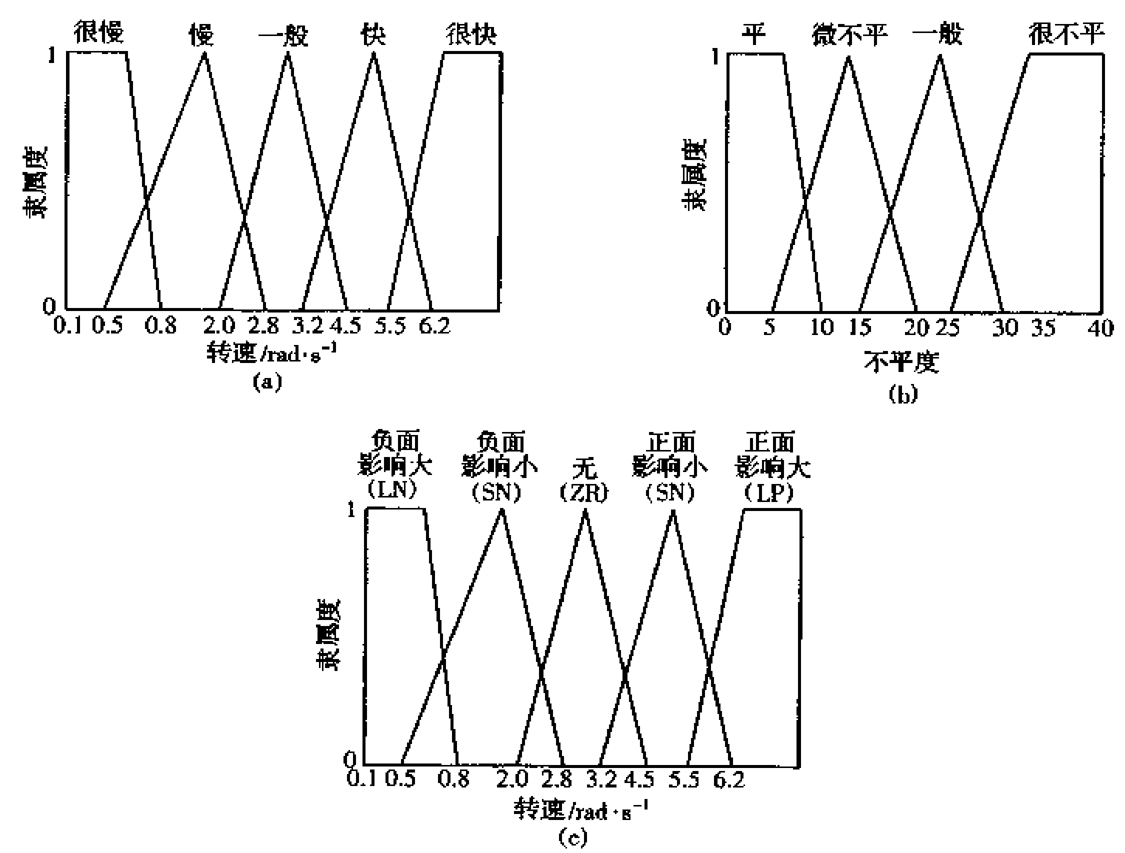


图 8.25　输入与输出的模糊集

其次，用模糊的陈述来构造模糊控制规则. 由速度与路面不平度的相互作用，可以构造出一个 4×5 的矩阵以表示在各种情况下油门应处的位置，其结果见表 8.8. 它把模糊控制器的输入与输出联系起来，这种表示也可以看做是另一种形式的规则，即：

if〈列的条件〉and〈行的条件〉then〈相交处即是动作〉

表 8.8　模糊控制规则表

转速表 / 路面平度	很慢	慢	一般	快	很快
平	LP	SP	ZR	SN	LN
微不平	LP	SP	ZR	ZR	SN
一般	LP	SP	SP	SP	ZR
较不平	LP	LP	LP	SP	SP

根据模糊控制规则，就可构造出相应的模糊神经网络，如图 8.26 所示.

注意这个系统类似于神经网络的形式，但其输入所连接的是不同的模糊控制规则，而不是普通神经网络的节点. 图 8.26 未画出精确量和模糊量的相互转换部分.

下面简要介绍一下自适应算法对模糊神经网络的修改. 例如模糊规则加权值的修改，这种修改算法是很简单的，它由各权值 W_i 乘以一个误差系数而组成. 在有导师指导的学习中，该系数是实际系统输出与正确输出值之比. 在自适应模式中，这个系数是从最优控制区的中心点到

系统实际响应区中心的均方距离.经过一段时间的训练后,将使贡献大的规则得到加强,而贡献小的规则受到削弱.同时,也使规则更加集中于输入数据的密集区.在这个例子中,开始就先令所有权值均为1,经过一段训练后,权值发生变化,如表8.9所示,系统的平衡点向右下方移动,即移向输入数据密集的区域.

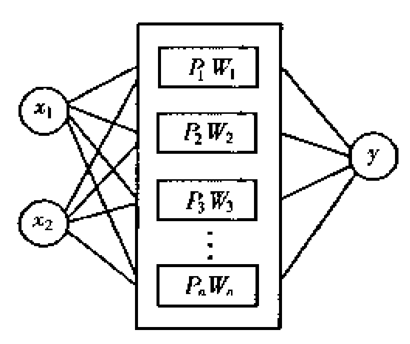


图 8.26　模糊神经网络

表 8.9　模糊控制规则表

路面平度＼转速表	很慢	慢	一般	快	很快
平	0.70	0.74	1.00	1.11	1.25
微不平	0.65	0.97	1.08	*1.20	*1.30
一般	0.82	0.94	*1.11	*1.28	*1.31
较不平	091	1.03	*1.18	*1.39	*1.44

表中带"*"号的是输入数据的密集区的部分.

对模糊神经网络实行自适应改变的另一种方法是模糊集论域宽度的改变.这种方法是根据反馈信号调整模糊集,使它变宽或变窄.也是根据实际输出与期望输出的误差来改变,但不同之处在于修改规则的加权值时,实际上只利用了上述误差的绝对值来进行调整.

8.3.2　适应性模糊联想记忆系统

一个模糊系统S将一簇模糊集合映射到另一簇模糊集合,即将相近的输入映射到相近的输出,这种行为称做联想记忆,该系统称为模糊联想记忆系统(Fuzzy Associative Memories System,简称FAM).最简单的模糊联想记忆编码是(A_i,B_i),表示p维模糊集合B_i与n维模糊集合A_i的关系.这与简单的神经网络功能相似,其差别是不需要适应性训练.模糊联想记忆系统中,规则库可写成$(A_1,B_1),\cdots,(A_m,B_m)$.每个输入$A$可以不同的程度作用FAM系统中的规则,FAM映射$A$到$B_i'$,$B_i'$是$B_1$的一部分.相应的输出模糊集合$B$可以通过各部分作用的模糊集合结果的组合得到,即

$$B = w_1^* B'_1 + w_2^* B'_2 + \cdots + w_m^* B'_m \tag{8.3.1}$$

其中w_i反映模糊联想(A_i,B_i)的可信度、频率或强度.最后,将输出模糊集合B通过去模糊变成确定的数值y_i,图8.27给出了FAM系统的一般结构.

将神经网络思想引入FAM系统构成的自适应模糊系统,称为适应性模糊联想记忆系统(Adaptive Fuzzy Associative Memories System,简称AFAM).它是一种随时间变化的模糊系统,系统参数随样本和处理的数据而逐渐变化,即通过神经网络的学习规则,如Hebb规则、Delta规则等,其基本形式是:

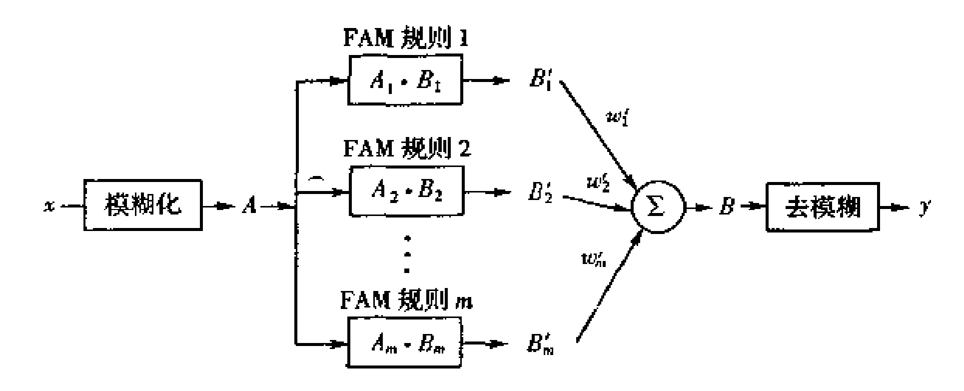


图 8.27 模糊联想记忆系统的一般结构

$$w_i(t+1) = \alpha \times w_i(t) + \Delta w_i(t+1) \tag{8.3.2}$$

8.3.3 基于神经网络的模糊系统建模

模糊系统建模(Fuzzy Modeling)是一种利用模糊推理描述复杂的非线性系统的特征的方法.基于神经网络的模糊系统建模是将神经网络的技术应用于模糊专家系统,其核心是用神经网络来实现模糊系统的输入/输出映射关系.其功能结构与一般的神经网络专家系统大体相同,而不同的是它要存贮及运用的模式或模式对是模糊模式或模糊模式对.因此,在实际采用神经网络技术的模糊专家系统中,需要重点考虑的问题是:如何实现模糊模式或模模模式对的模糊联想记忆存储及如何实现模糊模式或模糊模式对之间的模糊联想及模糊映射.

为实现模糊模式的存储及映射,亦即实现常规意义下的模模规则的存储与运用,可将前馈型神经网络引入模糊推理.选用前馈型神经网络来构造模糊推理系统,通常使用模糊穴-穴映射模型.基本思想是将输入变量空间进行模糊划分,形成有限个模糊穴组或的穴空间,将这一过程称为"模糊化"过程.然后将模糊化后的变量作为前馈型神经网络的输入.经过前馈型神经网络的学习和训练后,其输出还要进行"惟一化"操作,将此过程称为"非模糊化"过程.原理如下:

设 $\boldsymbol{X}$ 为网络输入变量空间,将 $\boldsymbol{X}$ 划分为若干个称为"模糊穴"的模糊子空间 $\boldsymbol{X}_i(i=1,2,\cdots,N)$,这些模糊穴满足如下条件:

① $\forall x \in \boldsymbol{X}, \exists 1 \leqslant j \leqslant N, \mu_{X_j}(x) > 0$

② $\forall x \in \boldsymbol{X}, \exists 1 \leqslant j,k \leqslant N, \mu_{X_j}(x) \wedge \mu_{X_k}(x) > 0$

③ $\forall x_1, x_2 \in X_i, \forall \lambda \in [0,1], \mu_{X_j}(\lambda x_1 + (1-\lambda)x_2) \geqslant \mu_{X_j}(x_1) \wedge \mu_{X_j}(x_2)$

变量空间 $\boldsymbol{X}$ 经过模糊划分后,就转化为由有限个模糊穴组成的穴空间.而 $\boldsymbol{X}$ 中的任一对象也就可以由这些模糊穴来表征.例如,对任一 $x \in \boldsymbol{X}$,有

$$CX(x) = \{CX_j(x) = \mu_{X_j}(x), j = 1,2,\cdots,N\} \tag{8.3.3}$$

其中 $CX(x)$ 称为 x 所对应的模糊穴向量.根据 Zadeh 的定义,模糊推理系统可由 $CY(y) = G(CX(x))$ 表征.其中 $CY(y)$ 和 $CX(x)$ 分别为系统输出及输入所对应的模糊穴向量,G 是从 x 到 y 的模糊映射,在输入 x 后,由 $CY(y)=G(CX(x))$ 可获得输出模糊向量

$$CY(y) = \{CY_1(y), CY_2(y), \cdots, CY_m(y)\} \tag{8.3.4}$$

进一步可用"面积中心法"将结果非模糊化:

$$y = \sum w_j \times CY_j(y) / \sum CY_j(y)$$

即

$$y = \frac{\sum w_j \times \mu_{Y_j}(y)}{\sum \mu_{Y_j}(y)} \tag{8.3.5}$$

其中 w_j 为 Y 上的模糊穴 Y_j 的特征点，它满足

$$\mu_{Y_j}(w_j) = 1, \mu_{Y_j}(w_j + y) = \mu_{Y_j}(w_j - y) \tag{8.3.6}$$

基于模糊穴概念的模糊穴-穴映射模型，模糊神经网络可形式化的表示为图 8.28.



图 8.28　基于模糊穴概念的模糊神经网络

以 BP 网为例，这一类模糊神经网络结构如图 8.29 所示. 其中 FL 为模糊化层，BP-NET 为前馈型神经网络层，DL 为非模糊化层. FL 层的每一个节点对应着输入变量空间的模糊穴，FL 层的输出就是输入变量所对应的模糊穴向量，而前馈型神经网络的输出对应的是系统输出 y 对于其空间 Y 中某个模糊穴的隶属程度. FL 层将系统输入转化为模糊穴向量，前馈型神经网络实现模糊映射，DL 层将前馈型神经网络输出的模糊穴向量转化为系统的确定输出. 对于此类模糊神经网络，用于模糊映射的神经网络需预先用样本模式进行学习训练，而不是整个网络一起训练.

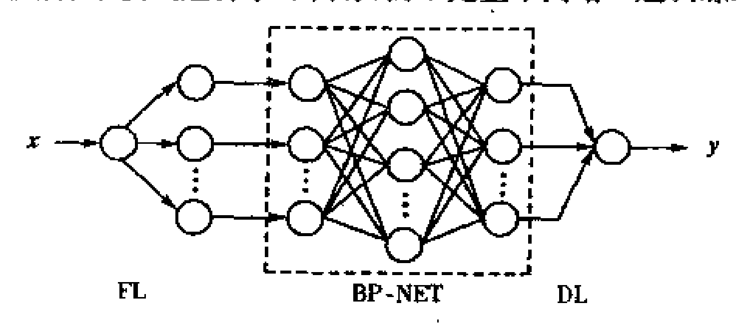


图 8.29　基于 BP 神经网络的模糊系统建模

进一步可将神经网络分别引入模糊化、模糊推理、解模糊判决模块，也即用三个神经网络分别模拟这三个模块的输入输出特性. 这类模糊神经网络结构如图 8.30 所示. 其中：A、B、C 层实现模糊化，D、E、F 层实现模糊推理，G、H、I 层实现非模糊化. C、D 层的节点数由输入变量的模糊集个数决定，F、G 层的节点数由输出变量的模糊集个数决定. 这种模糊神经网络比前者具有较高的精度.

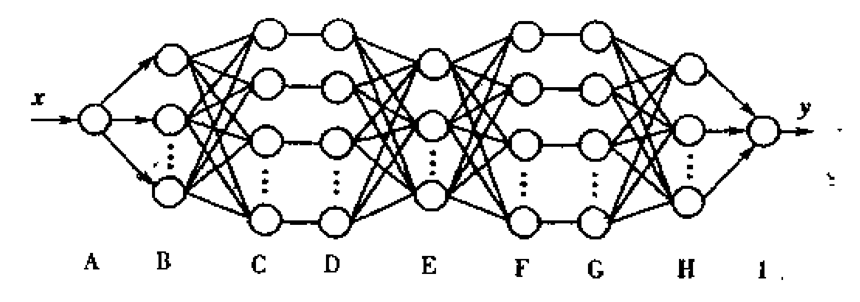


图 8.30　改进的基于 BP 神经网络的模糊系统建模

8.3.4　快速规则搜索的模糊系统建模

Matsushita S 等人曾提出一种快速模糊系统建模方法(Quick Fuzzy Modeling)，使得在建

立模糊系统之前，通过对输入数据进行粗略分类来搜索规则，然后基于所得到的规则建立模糊神经网络模型. 但该方法适用于二维输入空间，当输入空间大于二维时，输入子空间的相邻关系的确定比较复杂，而且输入的维数很大时，建模的层数也随之递增.

下面介绍一种不受输入维数限制的快速模糊规则搜索方法，并基于此方法进行模糊建模，用遗传算法训练网络，最后通过实验证明该方法的可行性和有效性.

一、快速模糊规则搜索方法

对于模糊系统而言，如果有 n 个输入变量，每个变量的论域上又定义 m 个模糊子集，那么总计共有 m^n 条模糊规则. 快速模糊规则搜索方法将从 m^n 条规则中抽取若干规则，从而简化了模糊系统的结构，减少了计算量.

网络结构如图 8.31 所示，由四层功能不同的节点构成. 各层节点的输入输出关系如下：

A 层：　　$Q_i = I_i = X_i$

B 层：　　$Q_i = \mu(I_i)$

C 层：　　$Q_i = \prod I_i$

D 层：　　从 C 层选择 K 个规则

其中，A 层为输入层，B 层实现输入量的模糊化，C 层实现模糊推理，即实现模糊规则前件的“与”操作，输出为样本对规则前件的适合度. D 层通过所有学习样本对每条模糊规则前件的适合度的大小，依照一定的算法，从中选取 K 条模糊规则，并将每条被选取模糊规则的前件存储在一个二维数组 RULES[K][n]中，其中，K 为规则条数，n 为输入变量的维数. 然后计算每条被选取模糊规则的后件，即规则的输出值，并将其存储起来.

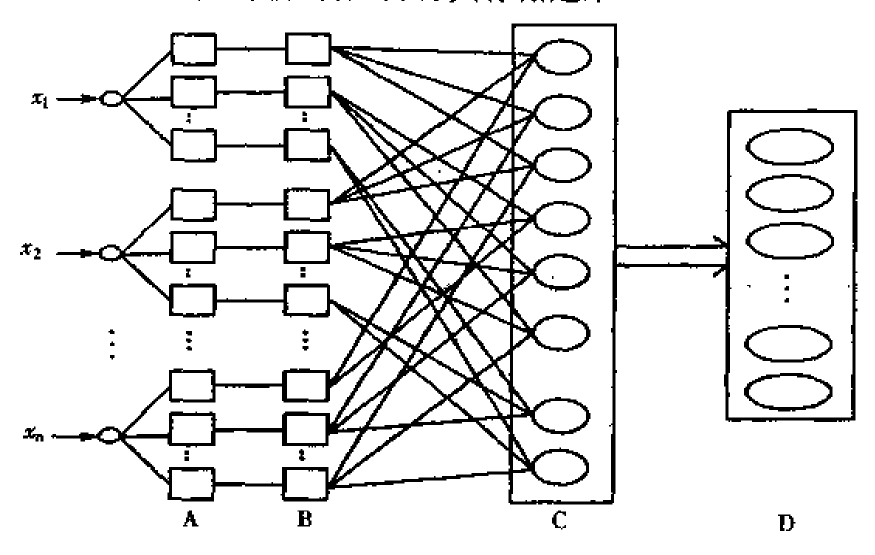


图 8.31　基于快速模糊规则搜索的 FNN 模型

通常，一个学习样本可以同时适合于多条模糊规则，即一个学习样本同时为多条规则的相关样本. 显然，一条模糊规则的相关样本的数目越大，说明此规则被选取的概率也就越高. 某一模糊规则有相关样本时，称此规则为有效规则，某一模糊规则没有相关样本时，称此规则为无效规则. 选取模糊规则的具体算法如下：

(1)按顺序从所有可能的模糊规则集中取一模糊规则.

(2)分别计算所有学习样本对此规则前件的适合度 fit，找出所有与此规则相关的样本，构成其相关样本集合. 当相关样本集合不为空集时，根据下式计算此规则的确定输出：

$$y^* = (\sum_{i=1}^{N} y_i \times \mathrm{fit}_i) / \sum_{i=1}^{N} \mathrm{fit}_i \tag{8.3.7}$$

其中，y^* 为该模糊规则对应的输出值，y_i为该规则前件的学习样本的确定输出值，N 为相关样本集合中的样本数.

(3)将此规则输出 y^* 模糊化.计算出此输出对输出模糊子空间的隶属度值(μ_1,μ_2,μ_3)，则此模糊规则的前件和后件为:if X_1 is…and X_2 is…and X_3 is…then Y is small/μ_1,is mid/μ_2,is big/μ_3.

(4)重复上述步骤，直到完成对所有可能模糊规则的计算.

(5)依据模糊规则的相关样本集合的大小，选取一定数量 K 条有效模糊规则，并将每条规则的前件和后件存储起来.

与 K 值的选取相关的因素包括:①输入变量的维数和每个输入变量的模糊子集数目，即所有可能的模糊规则数目.②模糊系统本身的特性.③决定相关样本的门限 α 值的大小.针对一个具体问题，在确定了输入维数和每个输入变量的模糊子集数后，该模糊系统的所有可能的模糊规则数目是一定的，那么 α 值越大，则所能得到的有效规则数就越少，反之，α 值越小，则所能得到的有效模糊规则数就越大.在确定了一个合适的 α 值以后，可以得到一个有效模糊规则集，其中包含 K 条模糊规则.由于在上述规则抽取过程中没有网络权值的学习训练过程，所以此过程非常迅速.

二、基于快速模糊规则搜索建立模糊神经网络模型

利用上述快速模糊规则抽取方法得到的有效模糊规则集，当用此模糊规则集所构成的模糊系统，仅是粗略地反应了真实系统，对真实系统的逼近程度尚不完全符合要求，还需要对该模糊规则集的输入和输出的隶属度函数，及各规则的相对重要程度进行进一步的调整和改进.因此，可以利用上述快速模糊规则抽取方法得到的模糊规则集，建立模糊神经网络，考虑采用 8.2.3 节介绍的模糊加权神经网络模型(WFNN)，结构如图 8.32 所示.

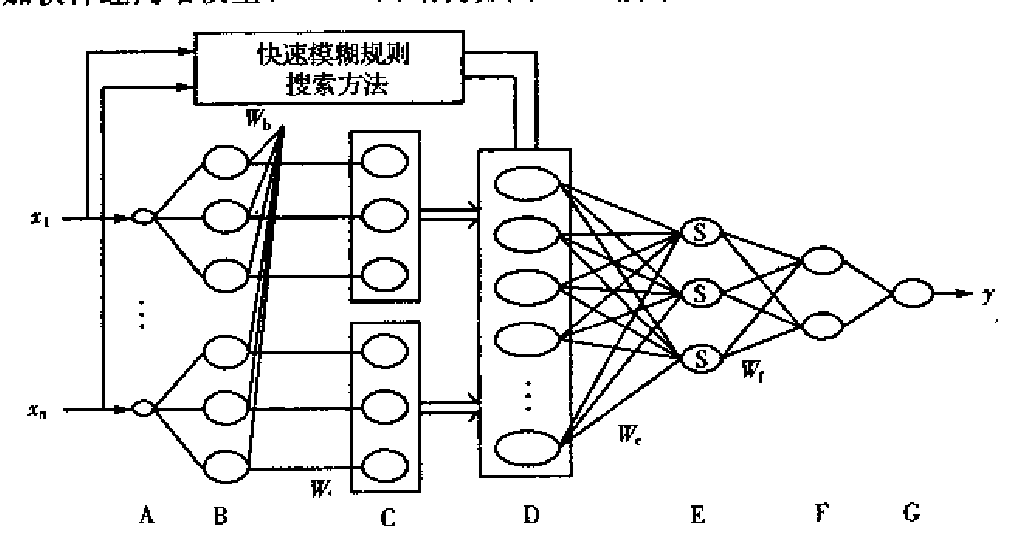


图 8.32　基于快速模糊规则搜索的 WFNN 模型

A 层为输入层，B、C 层实现输入变量的模糊化，选用 Sigmoid 型隶属度函数，D、E 层实现模糊推理，D 层实现模糊规则前提的'与'操作，E 层完成求和功能，给出规则的输出，F、G 层用"重心法"实现解模糊判决操作.首先，通过快速模糊规则搜索方法，得到 K 条有效模糊规则，可以确定该模糊神经网络模型的 D 层节点数为 K.其次，根据这 K 条规则的前件部分，还可以

确定 C 层和 D 层的连接方式. 例如，输入变量的模糊子集定义为{small，mid，big}时，模糊规则 R_i 的前件为：if X_1 is small and X_2 is small and X_3 is mid and X_4 is big and X_5 is mid then…，则对于 D 层第 i 个节点，即代表此规则的节点，其 5 个输入节点分别为 C 层的 C_0、C_3、C_7、C_{11}、C_{13}. 然后，根据模糊规则的后件部分，对权值 W_e 进行初始化，W_b 为可变阈值，$\boldsymbol{W}_c$、$\boldsymbol{W}_e$、$\boldsymbol{W}_f$ 为可变权值.

基于快速模糊规则抽取方法所建立的模糊神经网络模型，由于其节点和权值可以用隶属度函数和模糊规则解释，因此模糊规则集的输入、输出的隶属度函数，及各规则的相对重要程度需要通过学习训练进行调整和改进，实质上就是寻找合适的 $\boldsymbol{W}_b$、$\boldsymbol{W}_c$、$\boldsymbol{W}_e$、$\boldsymbol{W}_f$.

三、在金融预测中的应用

现在通过一个实际的股票开盘价的预测问题来验证此方法的实用性和有效性. 实验中使用上海市 1998 年 6 月～1998 年 12 月的每日股票开盘价，共 140 个数据. 用前 5 个数据作为输入，进行预测，得到下一个数据，这样连续 140 个数据可以产生 135 个输入输出模式，以其中前 100 个模式作为学习样本，其他的作为测试样本. 在本文中选取 m 为 3，n 为 5，模糊规则的总数为 243. 首先进行快速模糊规则的抽取，得到 82 条规则. 然后基于这 82 条模糊规则建立模糊神经网络. 采用遗传算法训练该模糊神经网络，训练 5000 代后，模糊神经网络输出结果与样本的输出的比较如图 8.33 所示. 从实验结果可以看出，网络误差在允许范围内，可见效果相当理想.

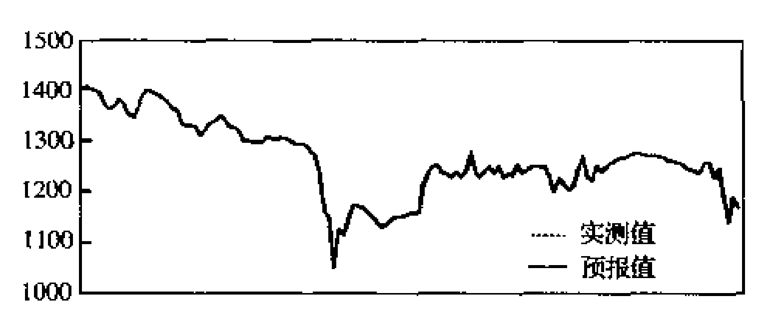


图 8.33 快速搜索 FNN 的实测值与预报值的比较

基于快速模糊规则搜索的模糊神经网络模型与传统的穷举所有模糊规则的模糊神经网络模型比较，从图 8.34 可以看出，前者较之后者，不仅在网络结构上更简洁，而且其网络权值优化的速度快. 原因是基于快速模糊规则搜索的模糊神经网络模型，通过快速模糊规则的接索，删掉了一些在训练过程中网络权值趋于零的规则，使得训练速度显著提高.

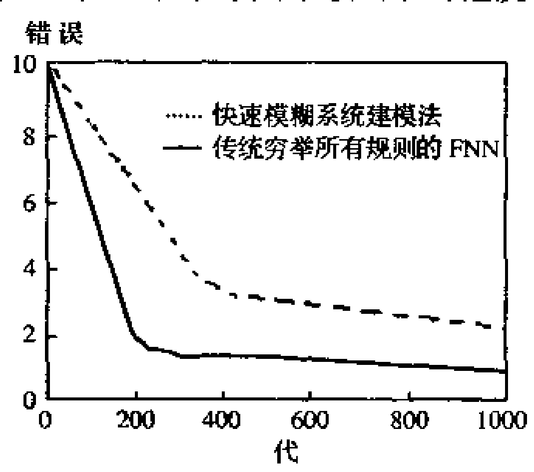


图 8.34 快速模糊系统建模与传统方法的比较

参 考 文 献

[1]Zadch L A. Fuzzy Sets. Information Control,1965,8:pp. 338~353

[2]Zadch L A. Fuzzy Algorithm. Information Control,1968,12:pp. 94~102

[3]Mamdani E H. Application of Fuzzy Algorithms for Simple Dynamic Plant. Proc. IEE, 1974,121,pp. 1585~1588

[4]张乃尧,阎平凡. 神经网络与模糊控制. 北京:清华大学出版社,1998年.

[5]章卫国,杨向忠. 模糊控制理论与应用. 西安:西安工业大学出版社,1999年.

[6]李人厚. 智能控制理论和方法. 西安:西安电子科技大学出版社,1999年.

[7]寺野 寿郎等. ファジィシステム入门. 东京:オーム社,1990年.

[8]Liang Yanchun,Wang Zheng,Yang Xiaowei,Zhou Chunguang. Identifi cation of Non-Linearities in Cushioning Packaging Using Neural Networks with Fuzzy Adaptive Control. Mechanics Research Communication. 1997,Vol. 24,No. 4,pp. 447~455

[9]Liang Yanchun,Gong Wenying,Zhou Chunguang. Idetification of Nonlinear Characteristics in Cushioning Packaging Using Evolutionary Neural Networks. Mechanics Research Communication,1998,Vol. 25,No. 4,pp. 395~403

[10]周春光,梁艳春,田勇,胡成全,孙新. 基于模糊神经网络味觉信号识别的研究. 算机研究与发展,1999,Vol. 36,No. 4,pp401~409

[11]梁艳春,王证,周春光. 模糊神经网络在时间序列预测中的应用. 计算机研究与发展,1998,Vol. 35,No. 7,pp651~665

[12]周春光,张冰,梁艳春,胡成全,常迪. 模糊神经网络及其在时间序列分析中的应用. 软件学报,1999,Vol. 10,No. 12,pp1303~1309

[13]田勇,周春光,梁艳春. 一种基于模糊加权型推理法的模糊神经网络. 小型微型计算机系统,1999,Vol. 20. No. 4,pp275~280

[14]吕慧英,郭东伟,周春光,梁艳春. 氨基酸序列中同源序列的识别. 小型微型计算机系统,2000,Vol. 21,No. 6,pp. 646~649

[15]权炜,周春光,梁艳春. 基于快速模糊规则搜索的模糊神经网络. 小型微型计算机系统,2001,Vol. 21. No. 01,pp74~80

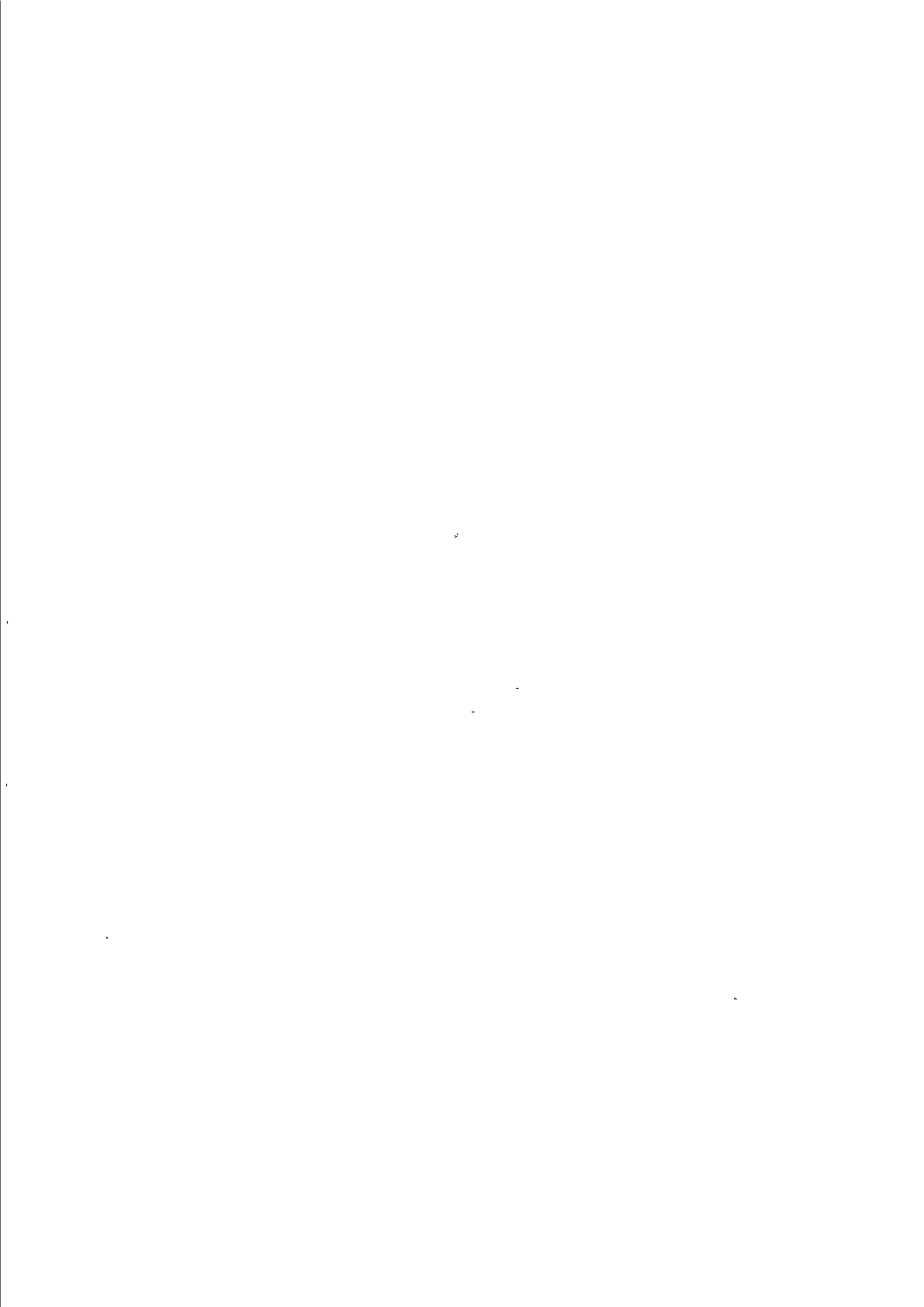
[16]田勇,周春光,梁艳春. 基于模糊神经网络味觉信号的识别研究. 计算机应用,1998,No. 1,pp. 42~48

[17]王政,梁艳春,周春光. 模糊神经网络在时间序列预测中的应用. 计算机应用,1998,No. 1,pp. 49~56

[18]梁艳春,杨晓伟,周春光. 基于模糊自适应BP算法的包装件缓冲层非线性特性识别. 包装工程,1996,Vol. 17,No. 4,pp. 9~12

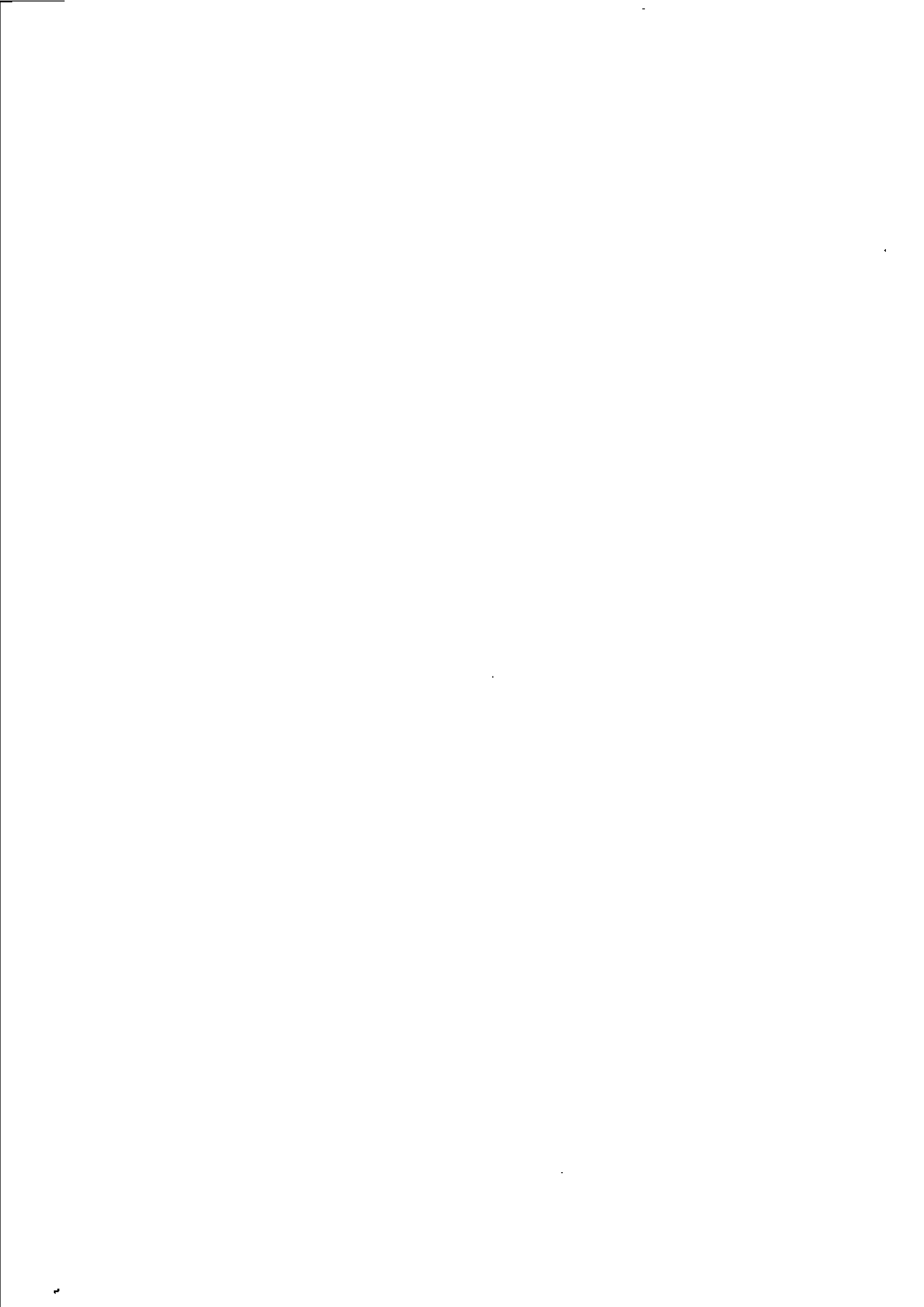
[19]梁艳春,冯大鹏,周春光. 非线形振动系统恢复力识别的模糊自适应BP算法. 力学应用与研究,大连:大连理工大学出版社,1999年6月,pp107~110

[20]Zhou Chunguang,Liang Yanchun,TianYong. A Fuzzy Neural Network Based on Fuzzy Weighted Reasoning Method. IWADS2000 The First International Workshop on Autonomous Decentralized System. Chengdu China:2000. pp190~195



第三篇

进化计算



第九章 遗传算法

遗传算法(Genetic Algorithm,GA)是一类以 Darwin 自然进化论和 Mendel 遗传变异理论为基础的求解复杂全局优化问题的仿生型算法,它是由美国 Holland J H 教授首次提出的. GA 基于适者生存,优胜劣汰的进化原则,对包含可能解的群体反复使用遗传学的基本操作,不断地生成新的群体,使种群不断进化,同时以全局并行搜索技术来搜索优化群体中的最优个体,以求得满足要求的最优解或准最优解. GA 可广泛用于组合优化、机器学习、自适应控制、规划设计、人工生命、人工神经网络训练和图像处理等领域,是 21 世纪有关智能计算中的关键技术之一. 本章简要介绍生物进化的概念,示例介绍传统遗传算法的设计方法和步骤,简要介绍遗传算法的研究历史,发展现状,以及遗传算法当前研究的热点与待解决的问题.

§9.1 生物进化与遗传算法的发展

一、从生物进化到遗传算法

建立在 Darwin 进化论和 Mendel 遗传变异规律基础上的现代生物学认为,生物进化是从低级向高级,从简单向复杂,趋势向上,而又呈现出多枝并进,多样化发展的演化过程. 生物进化的原因自古至今有各种不同的解释,其中被人们广泛接受的是 Darwin 的自然选择学说.

自然选择学说认为,生物要生存下去,就必须进行生存斗争. 生存斗争包括种内斗争,种间斗争以及生物与无机环境之间的斗争. 在生存斗争中,具有有利变异(mutation)的个体容易存活下来,并且有更多的机会将有利变异传给后代;具有不利变异的个体容易遭淘汰,产生后代的机会也少得多. 因此,凡是在生存斗争中获胜的个体都是对环境适应性比较强的. Darwin 把这种在生存斗争中适者生存,不适者淘汰的过程叫做自然选择. Darwin 的自然选择学说表明,遗传和变异是决定生物进化的内在因素. 遗传是指父代与于代之间在性状上存在的相似现象. 变异是指父代与于代之间,以及子代的个体之间,在性状上或多或少地存在的差异现象. 在生物体内,遗传和变异的关系十分密切. 一个生物体的遗传性状往往会发生变异,而变异的性状有的可以遗传. 遗传能使生物的性状不断地传送给后代,因此保持了物种的特性,变异能够使生物的性状发生改变,从而适应新的环境而不断地向前发展. 变异是生物进化的根本保证,生物进化是遗传与变异相互作用的结果.

生物的各项生命活动都有它的物质基础,生物的遗传和变异也是这样. 现代细胞遗传学与分子遗传学的研究表明,遗传物质的主要载体是染色体(chromosome),染色体主要由 DNA(脱氧核糖核酸)和蛋白质组成,其中 DNA 又是最主要的遗传物质. 基因(gene)是有遗传效应的片段,它储存着遗传信息,可以准确地复制(reproduction),也能够发生突变,并通过控制蛋白质的合成而控制生物的性状. 生物体自身通过对基因的复制和交叉(crossover,即基因分离,基因自由组合和基因连锁互换)的操作使其性状的遗传得到选择和控制. 同时,通过基因重组,基因变异以及染色体在结构和数目上的变异产生丰富多彩的变异现象. 根据 Darwin 进化论,多种多样的生物之间所以能够适应环境而得以生存进化,是与上述的遗传和变异生命现象分

不开的.生物的遗传特性,使生物界的物种能够保持相对的稳定,而生物的变异特性,使生物个体产生新的性状,决定了生物体的多样性,以至于形成了新的物种,推动了生物的进化和发展.

生命科学与工程科学的相互交叉,相互渗透和相互促进是近代科学技术发展的一个显著特点,而遗传算法的蓬勃发展正体现了科学发展的这一特征和趋势,遗传算法和人工神经网络都是将生物学原理应用于科学研究的仿生学理论成果,是近年来信息科学、人工智能与计算机科学的两大热门研究领域.

遗传算法和人工神经网络的产生都受到了自然界中信息处理方法的启发,但其来源并不相同.人工神经网络是在对以人脑为主要代表的生物神经系统的组织结构和行为特征进行研究的基础上提出的,它侧重于对人脑某些特定功能的模拟,其强调大量神经元之间的协同作用和通过学习的方法解决问题是人工神经网络的重要特征.遗传算法是从自然界生物进化机制获得启示的,它是模拟Darwin的遗传选择和自然淘汰的生物进化过程的计算模型,由美国 Michigan 大学的 Holland J H 教授于 1975 年首先提出的. Holland J H 教授和他的研究小组围绕遗传算法进行研究的宗旨有两个,一是抽取和解释自然系统的自适应过程,二是设计具有自然系统机理的人工系统,这些研究无论对自然系统还是对人工系统都是十分有意义的.

科学研究、工程实际与国民经济发展中的众多问题可归结为"极大化效益,极小化代价"这类典型模型.在人工智能领域中,有不少问题需要在复杂而庞大的搜索空间中寻找最优解或准最优解.像邮递员路径问题(TSP)和规划问题等组合优化问题就是典型的例子.在求解此类问题时,若不能利用问题的固有知识来缩小搜索空间则会产生搜索的组合爆炸.因此,研究能在搜索过程中自动获取和积累有关搜索空间的知识,并自适应地控制搜索过程,从而得到最优解或准最优解的通用搜索算法一直是令人瞩目的课题.遗传算法就是通过向自然学习,借鉴生物进化机制求解问题的一种有效的算法.它的主要优点在于其本质上的并行性、广泛的可应用性(对目标函数的性态无要求,特例可以没有明确的表达式)、算法的高度鲁棒性、简明性与全局优化性.尽管遗传算法本身在理论和应用方法上仍有许多待进一步研究的问题,但它作为可靠、有效的全局优化算法,在组合优化问题求解、自适应控制、规划设计、机器学习和人工生命等领域的应用中已显示了巨大的潜力和诱人的前景.

二、遗传算法的发展历史

遗传算法是在 20 世纪六七十年代由美国 Michigan 大学的 Holland J H 教授及其学生和同事发展起来的.虽然早在 20 世纪 50 年代初期,就有研究人员开始研究运用数字计算机模拟生物的自然遗传与自然进化过程,到 20 世纪 50 年代末期,已有一些这方面的学术论文发表,但是当时从事这方面研究的主要是一些生物学家,研究的目的主要是为了更深人地理解自然遗传与自然进化现象. 20 世纪六七十年代初,Holland 教授开始认识到生物的自然遗传现象与人工自适应系统行为的相似性,他认为不仅要研究自适应系统,还要研究与之相关的环境.因此他提出在研究和设计人工自适应系统时,可以借鉴生物自然遗传的基本原理,模仿生物自然遗传的基本方法. 1967 年,他的学生 Bagley J D 在博士论文中首次提出"遗传算法(Genetic Algorithms)"一词.此后,Holland 指导学生完成了多篇有关遗传算法研究的论文. 1971 年,Hollstien R B 在他的博士论文中首次把遗传算法用于函数优化.

1975 年是遗传算法研究历史上十分重要的一年.这一年 Holland 出版了他的著名专著《自然系统和人工系统的自适应》(*Adaptation in Natural and Artificial Systems*),这是第一本系统论述遗传算法的专著,因此有人把 1975 年作为遗传算法的诞生年. Holland 在该书中

系统地阐述了遗传算法的基本理论和方法，并提出了对遗传算法的理论研究和发展极其重要的模式理论(schema theory). 该理论首次确认了结构重组遗传操作对于获得隐并行性的重要性. 直到这时，人们才真正认识到遗传操作到底在干什么，为什么又干得那么出色，这对于以后陆续开发出来的遗传操作具有不可估量的指导作用. 同年，De Jong K A 完成了他的博士论文《一类遗传自适应系统的行为分析》(*An Analysis of the Behavior of a Class of Genetic Adaptive System*). 该论文所做的研究工作，可看做是遗传算法发展进程中的一个里程碑，这是因为，他把 Holland 的模式理论与他的计算实验结合起来. 尽管 De Jong 和 Hollstien 一样主要侧重于函数优化的应用研究，但他将选择、交叉和变异操作进一步完善和系统化，同时又提出了诸如代构(generation gap)等新的遗传操作技术. 可以认为，De Jong 的研究工作为遗传算法及其应用打下了坚实的基础，他所得出的许多结论，迄今仍具有普遍的指导意义.

进入 20 世纪 80 年代，遗传算法迎来了兴盛发展时期，无论是理论研究还是应用研究都成了十分热门的课题. 此后，遗传算法广泛应用于各种复杂系统的自适应控制以及复杂的优化问题中.

从 20 世纪 80 年代开始，有关遗传算法的研究和应用日益普遍. 1985 年，在美国召开了第一届遗传算法国际会议(International Conference on Genetic Algorithms, ICGA)，并且成立了国际遗传算法学会(International Society of Genetic Algorithms, ISGA)，以后每两年举行一次. 1989 年，Holland 的学生 Goldberg D E 出版了专著《搜索、优化和机器学习中的遗传算法》(*Genetic Algorithms in Search, Optimization, and Machine Learning*). 该书总结了遗传算法研究的主要成果，对遗传算法及其应用作了全面而系统的论述. 一般认为，这一时期的遗传算法从古典阶段发展到了现代阶段，该书则奠定了现代遗传算法的基础. 在欧洲，从 1990 年开始每隔一年举办一次 Parallel Problem Solving from Nature 学术会议，其中遗传算法是会议主要内容之一. 此外，以遗传算法的理论基础为中心的学术会议还有 Foundations of Genetic Algorithms，该会也是从 1990 年开始隔年召开一次. 这些国际会议论文，集中反映了遗传算法近些年来的最新发展和动向. 1991 年，Davis L 编辑出版了《遗传算法手册》(*Handbook of Genetic Algorithms*)，其中包括了遗传算法在工程技术和社会生活中的大量应用实例. 有关遗传算法的学术论文也不断在 *Artificial Intelligence*, *Machine Learning*, *Information science*, *Parallel Computing*, *Genetic Programming and Evoluable Machines*, *IEEE Transactions on Neural Networks*, *IEEE Transactions on Signal Processing* 等杂志上发表. 1993 年，MIT 出版社创刊了新杂志 *Evolutionary Computation*. 1997 年，IEEE 又创刊了 *Transactions on Evolutionary Computation*, *Advanced Computational Intelligence* 杂志即将发刊，由模糊集合创始人 Zadeh L A 教授为名誉主编. 目前，关于遗传算法研究的热潮仍在持续，越来越多的从事不同领域的研究人员已经或正在置身于有关遗传算法的研究或应用之中.

§9.2 传统遗传算法

遗传算法是模拟生物进化过程的计算模型，作为一种新的全局优化搜索算法，遗传算法以其简单通用，鲁棒性强，适于并行处理以及应用范围广等显著特点，日益受到各学科研究人员的普遍重视.

一、基本术语

遗传算法是生物遗传学和计算机科学相互结合、渗透而形成的一种新的计算方法. 在遗传算法中经常使用有关自然进化中的一些基础术语. 为了学习和应用遗传算法，了解这些术语是

必要的.

生物的遗传物质的主要载体是染色体,基因是控制生物性状的遗传物质的功能单位和结构单位.若干个基因组成染色体,染色体中的位置称为基因座(locus),而基因所取的值称为等位基因(alleles),基因和基因座决定了染色体的特性,也就决定了生物个体的性状.此外,染色体有两种相应的表示模式,即基因型(genotype)和表现型(phenotype).所谓表现型是指生物个体所表现出来的性状,而基因型指与表现型密切相关的基因组成.同一种基因型的生物个体在不同的环境条件下可以有不同的表现型,因此表现型是基因型与环境条件相互作用的结果.在遗传算法中,染色体对应的是数据或数组,在传统遗传算法中,这通常是由一维的串(string)的结构数据来表现的.串中位置(string position)对应上述的基因座,而各位置上所取的值对应上述的等位基因.遗传算法处理的是染色体,或者称为基因型个体(individual).一定数量的个体组成了群体(population).群体中个体的数目称为群体规模(population size),各个体对环境的适应程度称为适应度(fitness).

执行遗传算法时包含两个必要的数据转换操作,一是表现型到基因型的转换,二是基因型到表现型的转换.前者是把搜索空间中的参数或解转换成遗传空间中的染色体或个体,此过程称为编码(coding)操作;后者是前者的逆操作,称为译码(decoding)操作.

为便于在以后学习和研究遗传算法时查阅方便,将生物遗传学和遗传算法中所使用的一些基本术语的对应关系列于表 9.1.

表 9.1　生物遗传学与遗传算法基础术语对照表

生物遗传学(natural genetics)	遗传算法(GA)
染色体(chromosome)	串(string),个体(individual)
基因(gene)	特征(feature),个性(character)
等位基因(locus)	串中位置(string position)
基因型(genotype)	结构(structure)
	参数集(parameter set)
表现型(phenotype)	解码结构(decoded structure)
	候选解(alternative solution)
遗传隐匿(epistasis)	非线性(nonlinearity)

二、传统遗传算法

遗传算法是具有"生成+检测"(generate and test)的迭代过程的搜索算法,其基本处理流程如图 9.1 所示.

遗传算法是一种群体型操作,该操作以群体中的所有个体为对象.遗传算法的三个主要操作算子是选择(selection)、交叉(crossover)和变异(mutation),它们构成了所谓遗传操作(genetic operation),使遗传算法具备了其他传统方法所没有的特性.

遗传算法中包含如下五个基本要素:

①参数编码;

②初始群体的设定;

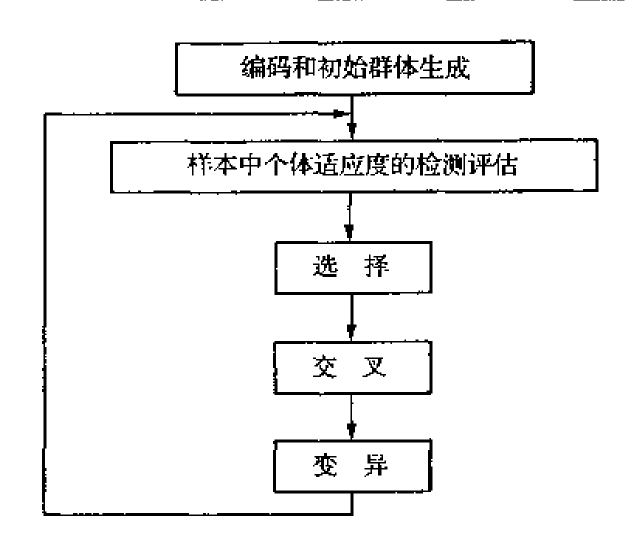


图 9.1　遗传算法的基本流程

③适应度函数的设计；

④遗传操作设计；

⑤操作参数设定(主要指群体规模以及执行遗传操作的概率等).

这五个要素构成了遗传算法的核心内容,我们将在第十章对其作详细论述.现在仅以一个简单的函数极值求解为例,来概述遗传算法的基本概念及迭代过程.

假定用遗传算法求函数 $f(x)=x^2$ 的最大值,设 $x\in[0,31]$,且 x 只取整数值.表 9.2 给出了用遗传算法求解该问题的计算过程和结果.

表 9.2　用遗传算法对 $f(x)=x^2$ 求极值的计算流程

串编号	初始群体(随机产生)	x值(无符号整数)	适应度 $f(x)=x^2$	选择概率 $p_i=f_i/\sum_j f_j$	适应度期望值 $f_i/\bar{f}$	实际计算(来自赌轮)	复制后配对库(竖线表示交叉点)	交叉位置(随机选择)	新一代群体	x值	适应度 $f(x)=x^2$
1	01101	13	169	0.14	0.58	1	0110|1	4	01100	12	144
2	11000	24	576	0.49	1.97	2	1100|0	4	11001	25	625
3	01000	8	64	0.06	0.22	0	11|000	2	11011	27	729
4	10011	19	361	0.31	1.23	1	10|011	2	10000	16	256
总和			1170	1.00	4.00	4					1754
平均			293	0.25	1.00	1					439
最大			576	0.49	1.97	2					729

下面针对表 9.2 中的各计算量详细说明如下(表中 $\bar{f}$ 为群体的平均适应度)：

1. 编码

由于遗传算法不能直接处理解空间的解数据,所以必须通过编码将其表示成遗传空间的基因型串结构数据.在表 9.2 中,将变量 x 编码为 5 位长的二进制无符号整数形式.比如 $x=19$ 可表示为 10011 的形式.

2. 初始群体的形成

由于遗传算法执行群体型操作,所以必须为遗传操作准备一个由若干个属于解空间的初始解组成的初始群体.在表 9.2 中取群体规模为 4,即群体由 4 个个体组成,每个个体通过随机方法产生.

3. 适应度评估检测

遗传算法在搜索进化过程中一般不需要其他外部信息,仅用评估函数值来评估个体的优

劣，并以此作为后继遗传操作的依据. 评估函数值又称做适应度(fitness). 本例根据$f(x)=x^2$来评估群体中各个体. 为了获得适应度值，必须把基因型个体译码成表现型个体，即搜索空间中的解，比如 01101 要译码为 13.

$$\begin{aligned}(01101) &= 0\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0\\ &=8+4+1\\ &=13\end{aligned}$$

4. 选择(selection)

选择或复制操作的目的是从当前群体中选出优良的个体，使它们有机会作为父代为下一代繁殖. 判断个体优劣的标准就是各自的适应度值. 这一操作借用了 Darwin 适者生存的进化原则，即个体适应度越高，其被选择作为父代的机会就越多. 选择操作的实现方式很多，这里采用与适应度值成正比的概率方法进行选择的方式，即以

$$P_i=\frac{f_i}{\sum_j f_j}$$

作为第 i 个个体的选择概率. 表 9.2 给出了选择 4 个个体的概率，由此概率可计算出每个个体期望被选择的次数. 也可以采用如图 9.2 所示的赌轮(roulette wheel)方式来决定各个体的选择次数. 赌轮上已经根据各个体适应度值按比例进行了分配. 表 9.2 给出了转动赌轮 4 次得到的相应的结果：个体 1 和个体 4 各复制一份，个体 2 复制 2 份，个体 3 不复制而被淘汰. 这正是我们所期待的结果，即最优秀的个体获得了最多的生存繁殖机会，最差的个体被淘汰. 由此得到的 4 份复制送到配对库(mating pool)以备配对繁殖.

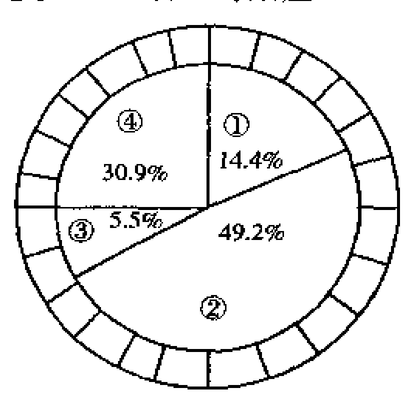


图 9.2　使用赌轮方法的选择

5. 交叉

简单的交叉(即单点交叉)可分两步进行：首先对配对库中的个体进行随机配对，然后在配对个体中随机设定交叉点，配对个体彼此交换部分信息. 如在表 9.2 中，配对库中的个体 2 与个体 1 配对，交叉点为 4，通过交叉得到两个新的个体：

$$\begin{matrix}0110|1 & \xrightarrow{\text{交叉}} & 01100\\ 1100|0 & & 11001\end{matrix}$$

交叉操作是遗传算法中的最主要的遗传操作. 通过交叉操作得到了新一代个体. 比较表 9.2 中新旧群体，可以发现新群体中个体适应度的平均值和最大值都有了明显提高. 由此可见，新群体中的个体确实是朝着所期望的方向进化了.

6. 变异

变异操作是按位(bit)进行的,即把某一位的内容进行变异.对于二进制编码的个体来说,若某位原为0,则通过变异操作就变成了1,反之亦然.变异操作同样也是随机进行的.一般而言,变异概率P_m都取得很小.在本例中,取$P_m=0.001$,由于群体中共有$20\times0.001=0.02$位可以变异,这意味着本例中群体内几乎没有一位可变异.变异操作是十分微妙的遗传操作,它需要和交叉操作妥善配合使用,目的是挖掘群体中个体的多样性,克服有可能陷于局部解的弊病.

在本例的模拟计算中,仅通过一代进化就使问题的解得到了优化.由表9.2可见,在新群体中,最大值和平均值特性都得到了改进.群体的平均适应度由293改进为439,最大适应度由576增加到729.如果按图9.1所示,进行多次迭代处理,最终一定可以得到最优解或近似最优解.

上述的遗传算法操作过程构成了传统的遗传算法,有时也称为简单遗传算法(simple GA,SGA).SGA的特点是:①采用赌轮选择方法;②随机配对;③采用单点交叉;④群体内允许有相同的个体存在.

通过上述简单例子的讨论可以看出,遗传算法所依据的两个必不可少而又最重要的操作(选择和交叉)是如此的简单可行.除了随机数产生、串结构复制以及部分串结构数据互换外,似乎没有更多更复杂的行为.

§9.3　遗传算法的特点与研究课题

一、遗传算法的特点

遗传算法的特点,可以从它和传统优化方法的对比中充分体现出来.

在遗传算法被广泛应用之前,优化问题的解法主要有以下几种:

(1)解析方法:通过求目标函数导数的零点或一系列迭代计算过程来求最优解.一般来说,若目标函数连续可微,解的空间方程比较简单,解析方法还是可以胜任的.但当方程组的变量较多时,无论解析求解还是迭代计算都是难度较大的.另外,对于如图9.3所示的多峰函数极值问题,这类方法容易陷入局部最优解,且要求目标函数有较好的连续性和可微性.

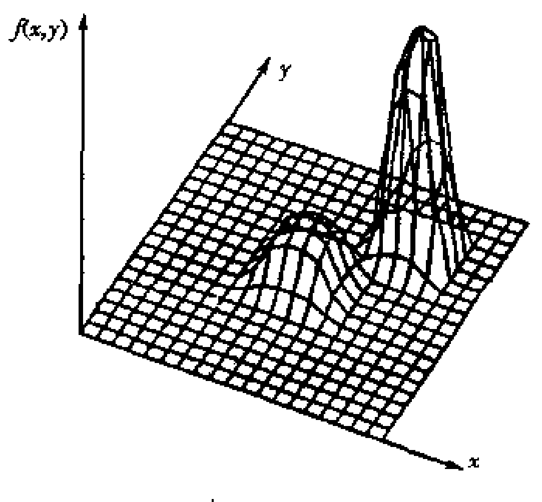


图9.3　多峰函数

(2)穷举法:在连续的有限搜索空间或离散的无限搜索空间中,计算并比较每一点的目标函数值,求出最优解.该方法简单易行,但当搜索空间大时,计算量的迅速增加使这类算法失效.

(3)随机搜索方法:这类算法主要有 Monte Carlo 法和模拟退火法. Monte Carlo 法盲目性大;模拟退火法在实际应用中较成功,但从一点到另一点的迭代过程使多峰问题易陷入局部最优解,且计算量过大.

对于一种优化算法寻求最优点不是惟一目的,实际经常退到的优化问题中,更重要的目标往往是进化,即优化过程应该是一个不断改进的过程,对于复杂系统的优化更是如此.遗传算法就是一种利用自然选择和进化思想在高维空间中寻优的方法,它不一定能寻得最优(optimal)点,但是它可以找到更优(superior)点,这种思路与人类行为中成功的标准是很相似的.遗传算法在运行过程中可能会暂时停留在某些非最优点上,直到变异发生使它跃居到另一个更优点上.遗传算法寻优过程的一个重要特点是它始终保持整个群体的进化.这样即使某个体在某时刻丧失了有用的特征,这种特征也可能会被其他个体所保留并连续发展下去.由于遗传算法仅须知道目标函数的信息,而不需要其连续可微等要求,因而具有广泛的适用性.同时它又是一种采用启发性知识的智能搜索方法,所以往往能在搜索空间高度复杂的问题上取得比传统优化方法更好的效果.

与传统优化方法比较,遗传算法主要有以下几个特点:

(1)遗传算法的处理对象不是参数(优化问题的参变量)本身,而是对参数集进行了编码的个体.这种编码操作,使得遗传算法可直接对结构对象进行操作(所谓结构对象泛指集合、序列、矩阵、树、图、链和表等各种一维或高维结构形式).这一特点使得遗传算法具有广泛的应用领域,例如:

通过对连接矩阵的操作,遗传算法可用来对神经网络或自动机的结构或参数加以优化.

通过对集合的操作,遗传算法可实现对规则集合或知识库的精练而达到高质量的机器学习目的.

通过对树结构的操作,遗传算法可得到用于分类的最佳结构树.

通过对任务序列的操作,遗传算法可用于任务规划,而通过对操作序列的处理,遗传算法可自动构造顺序控制系统.

(2)遗传算法的基本作用对象是多个可行解的集合,而非单个可行解.它是采用同时处理群体中多个个体的方法,即同时对搜索空间中的多个解进行评估.这一特点使遗传算法具有较好的全局搜索性能,减少了陷于局部优解的可能性.同时这又使得遗传算法本身具有良好的并行性.

(3)遗传算法仅用适应度函数值来评估个体,而无须搜索空间的知识或其他辅助信息.遗传算法的适应度函数不仅不受连续可微的约束,而且其定义域可以任意设定.对适应度函数的惟一要求是,对于输入可计算出能够进行比较的输出.遗传算法的这一特点使它的应用范围极大拓宽,使之可广泛应用于目标函数不可微、不连续、非规划、极其复杂或无解析表达式等类优化问题.

(4)遗传算法不是采用确定性规则,而是采用概率的变迁规则来指导它的搜索方向.遗传算法执行选择、交叉、变异等类似生物进化过程的简单随机操作,具有极强的鲁棒性.需要指出,遗传算法采用概率仅仅是作为一种工具来引导其搜索过程朝着搜索空间的更优的解区域

移动.因此尽管看起来它是一种盲目的搜索方法,但实际上有明确的搜索方向.

由于遗传算法具有上述操作简单、鲁棒性强、易于并行化、整体优化性强等特点,使得它在为数众多的领域得到了越来越广泛的应用.

二、遗传算法的研究课题

作为一种搜索算法,遗传算法的基本框架已经形成,在各种问题的求解和应用中展现了它的特点和魅力,同时也暴露出了在理论和应用技术上的许多不足和缺陷.应该指出,尽管不断有关于遗传算法的各种新策略和新提案提出,但它们几乎都是针对特定的问题求解而言的,对它们的评价也都是基于对比实验,其中缺乏深刻而具普遍意义的理论分析.正因为如此,遗传算法现阶段研究的重点又回到了基本理论的开拓和深化,以及更通用、有效的操作技术和方法的研究上.下面概述遗传算法现阶段研究课题的几个主要方面.

1. 遗传算法收敛性的研究

遗传算法源于自然选择和生物遗传学,相对于其鲜明的生物基础,遗传算法的理论基础公认是不完善的.这种不完善主要表现在缺少广泛而完整的有关遗传算法的收敛性理论.由于遗传操作的随机性,使得人们难于将传统的遗传算法纳入恰当的便于分析的数学模型,因此使得对于传统的遗传算法的收敛性分析成为十分困难的问题.目前关于遗传算法收敛性的研究都是在对于算法增加某些限制条件后得到的遗传算法的各种修正形式的基础上进行的.对于遗传算法收敛性的研究与应用研究相比,可谓甚少.关于遗传算法收敛性的研究是进化计算理论研究的一个重要课题,它对于改进遗传算法以及对于遗传算法的应用研究都具有重要的理论意义和指导意义.但是迄今为止,人们还未曾得到一个关于遗传算法的完整的收敛性分析结果.特别是关于各种改进的遗传算法的收敛速度估计是当前遗传算法理论研究中的一个难点,目前尚无任何结果,这也是当前最迫切需要解决的问题之一,因为它能从理论上对遗传算法的任何修正形式提供评判标准,以指明改进遗传算法效能的正确方向.

2. 优化搜索方法的研究

迄今为止,优化问题的求解仍在遗传算法研究中占很大比重,诸如 TSP 等组合优化问题一直是遗传算法十分活跃的研究课题.据报道,遗传算法对 431 个城市的 TSP 问题的求解已取得最优解,对 666 个城市已取得准最优解.尽管遗传算法比其他传统搜索方法有更强的鲁棒性,但它更善长全局搜索,而局部搜索能力却不足.研究发现,遗传算法可以用极快的速度达到最优解的 90% 左右,但要达到真正的最优解则要花费很长的时间.一些对比实验还表明,如果兼顾收敛速度和解的品质两个指标,单纯的遗传算法未必比其他搜索方法更优越.为此,除了要进一步改进基本理论和方法外,还要采用和人工神经网络、模拟退火技术、专家系统以及混沌理论等其他方法相结合的策略.许多研究结果表明采用这种混合模型可以有效地提高遗传算法的局部搜索能力,从而进一步改善其收敛速度和解的品质.

3. 学习系统的遗传算法研究

基于遗传算法的机器学习是当前遗传算法研究的一个重要内容.这一新的研究课题把遗传算法从历来离散的搜索空间的优化搜索算法扩展到具有独特的规则生成功能的崭新的机器学习算法.这一新的学习机制对于解决人工智能中的知识获取和知识优化精练的瓶颈难题带来了希望.

4. 遗传算法的并行分布处理

随着遗传算法应用的深入发展,并行分布遗传算法及其实现的研究越来越为人们所重视.

遗传算法由于执行群体性操作,所以本质上具有很好的并行分布处理特性.尤其是遗传算法中各个体的适应度计算(这是遗传算法中最大的开销)可独立进行而彼此间无需任何通信,所以并行处理效率是很高的.但是,标准的遗传算法除适应度计算外,几乎所有的遗传操作都是建立在全局统计处理的基础上,这意味着在整个进化过程中,这种全局统计处理所引起的通信开销依然是不可忽视的.因此,设计有效的并行遗传算法及相应的实现系统对遗传算法的理论研究和应用研究都是十分重要而迫切的任务.这一研究不仅是对遗传算法本身的发展而且对于新一代智能计算机体系结构的研究都是十分重要的.

5. 人工生命与遗传算法的研究

近几年来,通过计算机模拟,再现种种生命现象以达到对生命更深刻理解的人工生命的研究正在兴起.人工生命所涉及的生命现象包括生命的起源、自我增殖、自适应遗传进化和免疫等.遗传算法与人工生命这一崭新的研究领域正在不断渗透.目前已有一些学者对生态系统的演变、食物链的维持以及免疫系统的进化等,利用遗传算法作了生动的模拟.由于实现人工生命的手段很多,所以尽管基于遗传算法的进化模型是研究人工生命的主要基础理论之一,但遗传算法在实现人工生命中的基本地位和能力究竟如何,是值得研究的课题.

6. 遗传算法和进化规划、进化策略的比较与结合

进化规划(Evolutionary Programming,EP)和进化策略(Evolution Strategies,ES)几乎是和遗传算法同时独立发展起来的,同遗传算法一样,它们也是模拟自然界生物进化机制的智能计算方法,既同遗传算法具有相同之处也有各自的特点.进入20世纪90年代,这三种方法才开始彼此交流.目前,关于这三种方法之间的比较研究和彼此结合的探讨正在形成热点.

第十章　遗传算法的数学基础

遗传算法是一个以适应度(优化问题的目标函数)为依据,通过对群体的个体进行遗传操作来实现群体内个体结构重组的迭代处理过程.群体中的个体在这一迭代过程中,一代一代地得以优化并逐步逼近最优解.遗传算法具有鲜明的生物学背景,它模拟的是自然界生物优胜劣汰的进化过程.但是它的本质内涵是什么?它所依赖的基本遗传操作,即选择、交叉和变异算子为什么能使遗传算法具有传统优化方法所不具备的鲁棒性、自适应性和全局优化等特点?本章试图通过对群体中个体的结构模式的分析来回答这些问题.

§10.1　模式(Schema)概念

在第九章求解 $f(x)=x^2$ 极值的例子中,我们可以初步观察到:群体中个体的适应度越高,其生存的机会就越多,而通过交叉操作,在下一代中产生了适应度更高,即性能更好的个体.在由表 9.2 所描述的两代进化过程中,第一代的最优个体(11000)和次优个体(10011)经随机配对并在位置 2 进行交叉后,产生了新的更优的个体(11011).进一步观察则会发现,新的个体的结构模式(11011)与其父代个体(11 * * *)和(* * * 11)的结构模式之间似乎有着某种联系:①它们的模式结构有某种相似性;②这些相似模板(similarity templates)对应高适应度值(高子群体的平均适应度).由此,可以得到一种感性认识:遗传算法在搜索过程中一直在搜索群体中个体的某种重要的结构相似性.

为了描述群体中个体结构的相似性,从而揭示遗传算法的实质,引入了模式(schema)的概念.模式是一个描述字符串子集的串(又称为模板),在该集合中的串的某些位置上存在相似性.

不失一般性,考虑二值字符集{0,1},由此可产生通常的 0,1 字符串.现在再增加一个符号"*",称为无关符(don't care)或通配符,即"*"既可以被当做 0,也可以被当做 1.于是将二值字符集{0,1}扩展到三值字符集{0,1,*}.

定义 10.1　基于三值字符集{0,1,*}所产生的能描述具有某些结构相似性的 0,1 字符串集的字符串称为模式.

以长度为 5 的串为例,模式 *0100 描述了在位置 2、3、4、5 具有形式"0100"的所有字符串,即{00100,10100};模式 * * 0 * 1 描述了在位置 3 为 0,在位置 5 为 1 的所有字符串,即{00001,00011,01001,01011,10001,10011,11001,11011};模式 10101 描述了只有一个串的集合{10101}.

由此可见,模式的概念提供了一种简洁的用于描述在某些位置上具有结构相似性的 0、1 字符串集合的方法.需要指出,通配符"*"只是一个描述符,它仅仅是为了描述上的方便而引入的符号,而并非遗传操作中实际的运算符号.

模式概念的引入,并非只是为了描述上的方便.在引入模式概念前,所看到的遗传算法是:在某一代中,n 个相互独立的串在选择、交叉、变异等遗传算子作用下产生下一代的 n 个新的

相互独立的串.因为这些串都是相互独立的,互不联系的,所以并不清楚在两代之间究竟保留了什么性质,破坏了什么性质.而引入模式概念后,可以看到,一个串实际上隐含着多个模式,而一个模式又可以隐含在多个子串中.因为在串的每一位置上可以取它的实际值或通配符,所以,长度为 l 的串隐含着 2^l 个模式,规模为 n 的群体隐含着 $2^l \sim n \cdot 2^l-(n-1)$ 个模式,其具体的模式数目取决于群体中个体的多样性.例如,串 01 隐含着 2^2 个模式 01,0 *,* 1,* *;串 110 隐含着 2^3 个模式 110,11 *,1 * 0,* 10,1 * *,* 1 *,* * 0,* * *;群体{01,10}隐含着 7 个模式 01,0 *,* 1,* *,10,1 *,* 0.遗传算法中串的运算实际上是模式的运算.因此,通过分析模式在遗传操作下的变化,就可以了解什么性质被延续,什么性质被抛弃,从而把握遗传算法的实质.

为了讨论模式的运算,需要了解模式的基本性质.为此,下面给出模式的两个重要概念:模式阶(schema order)和定义长度(defining length).

至此可知,一个串中隐含着多个不同的模式,而不同的模式所匹配的串(称为模式的样本)的个数是不同的.比如模式 10 * * 1 * 与 1 * * * * * 相比,前者所匹配的样本的个数比后者少,即前者的确定性高.为了反映这种确定性的差异,我们引入模式阶的概念.

定义 10.2　模式 H 中的确定位置的个数称为该模式的模式阶,记为 $O(H)$.

例如,模式 1 * 01 * * 的阶数为 3,而模式 1 * * * * * 的阶数为 1.显然一个模式的阶数越高,其样本数就越少,因此确定性也越高.

但是模式阶并不能反映模式的所有性质.以后将会看到,即使具有同阶的模式,在遗传操作下也会有着不同的性质.为此,再引入定义长度的概念.

定义 10.3　模式 H 中第一个确定位置和最后一个确定位置之间的距离称为该模式的定义长度,记为 $\delta(H)$.

例如,模式 1 * 01 * * 的定义长度为 3,而 1 * * * * * 的定义长度为 0.

模式阶和定义长度描述了模式的基本性质.有了这两个概念,就可以开始讨论模式在遗传操作下的变化.

§10.2　模式定理

首先讨论选择操作对模式的作用.令 $A(t)$ 表示第 t 代中串的群体,n 为群体规模,设群体中的个体两两互不相同.以 $S_j(j=1,2,\cdots,n)$ 表示一代中第 j 个个体串,设串的长度为 l.假设在第 t 代,群体 $A(t)$ 中模式 H 所能匹配的样本数为 m,记为 $m(H,t)$.设选择操作以概率

$$P_i = \frac{f_i}{\sum_{j=1}^{n} f_j} \tag{10.2.1}$$

进行,其中 f_i 是个体 S_i 的适应度.不妨设第 t 代群体 $A(t)$ 中模式 H 所能匹配出的串为 S_i,其适应度为 $f_i(i=1,2\cdots,m)$,其中第 i 个串在第 $t+1$ 代被选择的概率如(10.2.1)式所示,于是其在第 $t+1$ 代期望被选择到的个数为 $m_i=nP_i$.从而 H 在第 $t+1$ 代中的样本数 $m(H,t+1)$ 为:

$$m(H,t+1) = \sum_{i=1}^{m} m_i = n\sum_{i=1}^{m} P_i = n\sum_{i=1}^{m} \frac{f_i}{\sum_{j=1}^{m} f_j} =$$

$$\frac{m(H,t+1)n\sum_{i=1}^{m}\frac{f_i}{m(H,t)}}{\sum_{j=1}^{n}f_j}=$$

$$m(H,t)n\frac{f(H)}{\sum_{j=1}^{n}f_j} \qquad (10.2.2)$$

其中

$$f(H)=\frac{\sum_{i=1}^{m}f_i}{m(H,t)}$$

为模式 H 在第 t 代中所有样本的平均适应度. 令群体平均适应度为

$$\bar{f}=\sum_{j=1}^{n}\frac{f_i}{n}$$

则有

$$m(H,t+1)=m(H,t)\frac{f(H)}{\bar{f}} \qquad (10.2.3)$$

可见,模式的增长(减少),即样本数的增加(减少),依赖于模式的平均适应度与群体平均适应度之比,那些平均适应度高于群体平均适应度的模式将在下一代中得以增长;而那些平均适应度低于群体平均适应度的模式将在下一代中减少.

假定模式 H 的平均适应度始终保持高于群体平均适应度,设高出部分为 $c\bar{f}$其中 c 为常数,于是有

$$m(H,t+1)=m(H,t)\frac{\bar{f}+c\bar{f}}{\bar{f}}=m(H,t)(1+c) \qquad (10.2.4)$$

设 $t=0$ 开始,c 保持常值,于是可得如下公式:

$$m(H,t)=m(H,0)(1+c)^t \qquad (10.2.5)$$

由上式可见,在选择算子的作用下,平均适应度高于群体平均适应度的模式将呈现指数级增长,而平均适应度低于群体平均适应度的模式将呈指数级减少.

注意到仅有选择操作,并不能产生新的个体,即不能对搜索空间中新的区域进行搜索,因此引入了交叉操作. 下面讨论在交叉算子作用下模式的变化,仅考虑单点交叉的情形.

为了考察在交叉算子作用下,哪些模式受影响,哪些模式没有受到影响,不妨考虑一个长度为 7 的串以及隐含于其中的两个模式:

$$\begin{aligned} S &= 0\ 1\ 1\ 1\ 0\ 0\ 0 \\ H_1 &= *\ 1\ *\ *\ *\ *\ 0 \\ H_2 &= *\ *\ *\ 1\ 0\ *\ * \end{aligned}$$

假定 S 被选中进行交叉,而交叉点等概率产生,即交叉点落在 1、2、3、4、5、6 的概率相同. 不妨假设交叉点为 3,即交叉发生在位置 3 和位置 4 之间,并以分割符"|"表示交叉位置,即

$$\begin{aligned} S &= 0\ 1\ 1\ |\ 1\ 0\ 0\ 0 \\ H_1 &= *\ 1\ *\ |\ *\ *\ *\ 0 \\ H_2 &= *\ *\ *\ |\ 1\ 0\ *\ * \end{aligned}$$

此时与 S 进行交叉的串(称做配偶)除了在模式的确定位与 S 相同(暂且不考虑这种可能

性)者外,模式 H_1 将遭到破坏. 因为位置 2 的"1"和位置 7 的"0"在交叉产生的子代个体中将被替代为不同的值. 例如,S 的配偶为 $S'=1000111$,则产生的后代个体为 $S_1=0110111$,$S_2=1001000$,它们都不是 H_1 的样本,即在交叉操作后,模式 H_1 丢失了. 而此时 H_2 的样本却仍然存在. 实际上无论 S 的配偶为何种串,H_2 中确定位 4 的"1"和确定位 5 的"0"都将一起传入子代. 当然,如果交叉点在位置 4,那么 H_2 也会遭到破坏. 由此例不难看出,由于交叉点等概率产生,模式 H_1 遭破坏的概率(在位置 2,3,4,5,6 交叉都遭破坏)远远超过模式 H_2 遭破坏的概率(只在位置 4 交叉遭破坏),即 H_2 的生存能力要强于 H_1.

为了定量化上述观察,注意到模式 H_1 的定义长度为 $\delta(H_1)=5$. 当交叉点在 $l-1=7-1=6$ 个位置上等概率随机产生时,模式 H_1 遭破坏的概率为 $P_d=\delta(H_1)/(l-1)=5/6$,即其生存概率为 $P_s=1-P_d=1/6$,而模式 H_2 的定义长度为 $\delta(H_2)=1$,其遭破坏的概率为 $P_d=\delta(H_2)/(l-1)=1/6$,其生存概率为 $P_s=1-P_d=5/6$.

一般来说,模式 H 只有当交叉点落在定义长度之外才能得以生存. 在单点交叉下,H 的生存概率 $P_s=1-\delta(H)/(l-1)$. 注意到交叉本身也是以一定的概率 P_c 发生的,因此模式 H 的生存概率为 $P_s=1-P_c\delta(H)/(l-1)$.

现在考虑先前暂且忽略的可能性,即交叉发生在定义长度内,模式 H 不被破坏的可能性. 在上面的例子中,如果 S 的配偶在位置 2、7 上有一位与 S 相同,则 H_1 将被保留. 由此可见,上面给出的生存概率只是一个下界,即考虑到各种可能性,应有

$$P_s \geqslant 1 - P_c \frac{\delta(H)}{l-1} \tag{10.2.6}$$

此式描述了 H 在交叉算子作用下的生存概率.

下面考虑模式 H 在选择和交叉算子的共同作用下的变化. 参照(10.2.3)式和(10.2.6)式,有

$$m(H,t+1) = m(H,t)\,\frac{f(H)}{\bar{f}}P_s \geqslant m(H,t)\,\frac{f(H)}{\bar{f}}\left[1 - P_c\frac{\delta(H)}{l-1}\right] \tag{10.2.7}$$

由(10.2.7)式可见,在选择和交叉算子的共同作用下,模式的增长取决于两个因素:模式的平均适应度是否高于群体平均适应度;模式是否具有短定义长度. 参照(10.2.5)式,高于群体平均遭应度且具有短定义长度的模式将呈指数级增长.

最后,考虑变异操作. 设串的某一位置发生变异的概率为 P_m,于是该位置不变的概率为 $1-P_m$,而模式 H 在变异算子作用下若不受破坏,则其中所有的确定位置(为 0 或 1 的位)必须保持不变,因此 H 保持不变的概率为 $(1-P_m)^{O(H)}$. 当 P_m 充分小($P_m \ll 1$)时,模式 H 在变异算子作用下的生存概率为

$$P_s = (1-P_m)^{O(H)} \approx 1 - O(H)P_m \tag{10.2.8}$$

由此得到,模式 H 在遗传算子选择、交叉和变异的共同作用下,其子代的样本数为

$$\begin{aligned} m(H,t+1) &\geqslant m(H,t)\,\frac{f(H)}{\bar{f}}\left(1 - P_c\frac{\delta(H)}{l-1}\right)(1-O(H)P_m) = \\ &m(H,t)\,\frac{f(H)}{\bar{f}}\left[1 - P_c\frac{\delta(H)}{l-1} - O(H)P_m + P_c\frac{\delta(H)}{l-1}O(H)P_m\right] \geqslant \\ &m(H,t)\,\frac{f(H)}{\bar{f}}\left[1 - P_c\frac{\delta(H)}{l-1}O(H)P_m\right] \end{aligned} \tag{10.2.9}$$

上式中忽略了二阶小量 $P_c\dfrac{\delta(H)}{l-1}O(H)P_m$. 通过此式，可以得到以下模式定理：

定理 10.4(模式定理)在遗传算子选择、交叉和变异的作用下，具有低阶、短定义长度并且平均适应度高于群体平均适应度的模式在子代中将得以指数级增长.

模式定理给出了经过遗传算子作用后，某一模式的样本在下一代中得以生存的数目的下界，它是遗传算法的理论基础. 为了更好地理解该定理，通过一个例子来分析遗传算法对模式的处理情况. 仍采用上一章中的求 $f(x)=x^2$ 极值的例子，初始群体的选择以及执行遗传操作的计算流程参见上章表 9.2.

考察 3 个模式：$H_1=1****$，$H_2=*10**$，$H_3=1***0$ 的运行情况. 有关遗传算法对模式的处理见表 10.1

表 10.1　遗传算法求 $f(x)=x^2$ 极值的模式处理

	选择前			选择后			下一代		
		串代表	模式平均适应度 $f(H)$	期望个数	实际个数	串代表	期望个数	实际个数	串代表
H_1	1 * * * *	2,4	468.5	3.20	3	2,3,4	3.20	3	2,3,4
H_2	* 1 0 * *	2,3	320	2.18	2	2,3	1.64	2	2,3
H_3	1 * * * 0	2	576	1.97	2	2,3	0.0	1	4

首先，考察模式 H_1 在遗传操作下的变化. 在选择阶段，各串按照其适应度在群体适应度中所占的比例进行复制. 由表 9.2 可见，只有串 2,4 是 H_1 的样本，由于串 2 的适应度大，其复制概率高，经过选择被复制成 2 份，而串 4 被复制成 1 份，即通过选择操作后，H_1 的样本增至 3. 由模式定理，模式 H_1 理论上应有 $m\dfrac{f(H_1)}{\bar{f}}$ 个样本，其中 $f(H_1)=\dfrac{567+361}{2}=468.5$，而 $\bar{f}=293$，因此 H_1 在选择算子作用下应有 $m(H_1,t+1)=2\cdot\dfrac{468.5}{293}=3.2\approx 3$ 个样本，这与实际遗传操作相符. 进一步观察则可以看到，由于 H_1 的定义长度为 0，所以交叉操作对 H_1 的样本没有影响. 而对于变异操作，在变异概率为 $P_m=0.001$ 的情况下，H_1 的 3 个样本遭到破坏的概率为 0.003，可以认为不发生变异. 由表 9.2 可见，子代中有 3 个 H_1 的样本 11001,11011,10000，即模式 H_1 的样本数确实像模式定理所预计的呈指数级增长.

下面再来考察模式 H_2 和 H_3 的变化. H_2 的初始样本数为 2，在选择作用下，样本数仍为 2. 这与模式定理所预计的 $2\cdot\dfrac{320}{293}=2.18\approx 2$ 一致. 而模式 H_3 的初始样本数为 1，预期值为 $1\cdot\dfrac{567}{293}=1.97\approx 2$，与实际值相符. 最后来看交叉操作，尽管 H_2 和 H_3 的阶数相同，选择后的样本数也相同，但由于两个模式的定义长度不同，所以在交叉算子作用下的结果也不同. 对于 H_2，由模式定理，其预期值应为 $m(H_2,t+1)=2.18\times\left(1-\dfrac{1}{4}\right)=1.64\approx 2$. 实际上，最终 H_2 的两个样本都得以保留. 而对于 H_3，其预期值为 $m(H_3,t+1)=1.97\times\left(1-\dfrac{4}{4}\right)=0$. 但事实上，$H_3$ 有一个样本得以保留. 这似乎与模式定理不相符合，但注意到模式定理给出的只是模式的样本在下一代中得以生存的下界. 事实上，H_3 之所以有一个样本能生存，是因为 H_3 的样本 11000 与

其配偶 10011，在 H_3 的确定位（位置 1、5）有一位相同. 对 H_2 和 H_3 的考察结果表明，由于 H_2 具有较短的定义长度，所以其生命力强于 H_3.

至此，我们通过一个例子验证了模式定理的正确性. 在遗传算子作用下，具有低阶、短定义长度、高平均适应度的模式得以指数级增长.

§10.3　关于模式定理的讨论

模式定理是遗传算法的重要理论基础之一，但是这一定理只给出了经过遗传算子作用后，某一模式在下一代中得以生存的样本数目的下界（(10.2.9)式）. 当仅有选择算子作用时，模式定理呈(10.2.3)式的等式形式，模式定理由等式变成不等式(10.2.9)的原因是交叉和变异运算. 本节讨论变异运算使模式定理由等式变成不等式的原因，给出在选择和变异算子作用下模式定理的精确表达式，并从理论上证明选择算子可以使整个群体中个体的质量得以改进.

对于给定的模式 H，其在第 t 代的群体 $A(t)$ 中的样本数为 $m(H,t)$，经选择运算后，H 的样本数变为 $m(H,t)\dfrac{f(H)}{\bar{f}}$，再经过变异运算，将使 H 的某些样本变为不属于模式 H 的个体，记这样的个体数为 m_d，同时变异操作也会使某些原来不属于 H 的个体变为模式 H 的样本，记这样的个体数为 m_g，于是有

$$m(H,t+1)=m(H,t)\frac{f(H)}{\bar{f}}-m_d+m_g \tag{10.3.1}$$

如果 m_d,m_g 可以求得，则(10.3.1)式便给出了在选择和变异作用下的模式定理的精确表达式. 为了便于与具有不等式形式的模式定理相比较，在(10.2.9)式中令 $P_c=0$，得

$$m(H,t+1)\geqslant m(H,t)\frac{f(H)}{\bar{f}}[1-O(H)P_m] \tag{10.3.2}$$

上式忽视了 m_g 项，即忽视了从变异运算中得到的使原来不属于 H 的个体变为 H 样本的那部分个体，从而仅给出了 $m(H,t+1)$ 的下界.

下面讨论关于 m_d,m_g 的计算. 当变异算子以概率 P_m 作用于一个个体时，也就是以 P_m 为概率相互无关地作用于该个体的每个字符. 设 S_j 是第 t 代群体中的一个个体，其在群体中所占的比例为 R_j，当 S_j 是模式 H 的样本（以下简记为 $S_j\in H$）时，经过变异操作可能使其变成与 H 有不同模式（以下简记为 $S_j\notin H$）的个体的概率为

$$P_{dj}=\sum_{i=1}^{O(H)}C_{O(H)}^{i}P_m^i(1-P_m)^{O(H)-i} \tag{10.3.3}$$

而群体中所有与 H 具有相同模式的个体所占的比例为 $\sum\limits_{j|S_j\notin H}R_j$，因此经变异运算后 H 中的样本被减少的概率为

$$P_d=\sum_{i=1}^{O(H)}[C_{O(H)}^{i}P_m^i(1-P_m)^{O(H)-i}]\sum_{j|S_j\in H}R_j \tag{10.3.4}$$

对于 $S_j\notin H$，经过变异后可能变成 $S_j'\in H$，下面计算这个概率值. 设个体 $S_j\notin H$，且有 i 位与 H 不同，于是变异后 $S_j'\in H$ 的概率为

$$P_{gj}=P_m^i(1-P_m)^{O(H)-i} \tag{10.3.5}$$

所以经过变异运算，从群体中的其他个体（原不属于 H 的个体）获得的与 H 具有相同模式的

个体的概率为

$$P_g = \sum_{i=1}^{O(H)} \left[P_m^i (1-P_m)^{O(H)-i} \sum_{j|S_j \notin H 且 S_j 与 H 有 i 位不同} R_j \right] \tag{10.3.6}$$

从而经选择和变异操作后，失去与获得的模式 H 的样本数分别为

$$\begin{cases} m_d = m(H,t) \dfrac{f(H)}{\bar{f}} P_d \\ m_g = m(H,t) \dfrac{f(H)}{\bar{f}} P_g \end{cases} \tag{10.3.7}$$

将其代人(10.3.1)式，即得到在选择和变异算子作用下模式定理的精确表达式.

下面利用仅有选择算子作用的模式定理(10.2.3)式来讨论遗传算法的平均适应度的渐近行为，以便于进一步理解选择算子在遗传算法中所起的重要作用.

设 F 为所求优化问题的可行域，$f(S)$ 为适应度函数，且 $0 < f(S) < +\infty$，并令

$$B_t = \{S \in F, f(S) \geqslant \bar{f}_t\} \tag{10.3.8}$$

其中 $\bar{f}_t = \dfrac{\sum_{i=1}^{n} f(S_i(t))}{n}$ 为第 t 代群体的平均适应度，于是有下述结论成立：

定理 10.5 若选择过程没有随机误差，则在第 t 代，选择运算后的任一个体 $S'(t)$ 属于 B_t 的概率高于选择前的任一个体 $S(t)$ 属于 B_t 的概率，即

$$P(S'(t) \in B_t) \geqslant P(S(t) \in B_t) \tag{10.3.9}$$

证明：设 $|B_t|$ 和 $|B'_t|$ 分别表示选择前后 B_t 中包含的个体的数目，将选择后 B_t 中的个体按照模式分为 q 类. 由于选择过程中没有新的个体生成，所以这 q 类模式在选择前已存在，以 $H_1(t), H_2(t), \cdots, H_q(t)$ 记之，并设它们在选择前后的样本数分别为 $m_i(H_i)$ 和 $m'_i(H_i)$ $(i=1, 2, \cdots, q)$. 由于选择过程没有随机误差，且 $0 < f(S) < +\infty$，所以选择前后集合 B_t 中的模式类完全相同. 于是利用模式定理(10.2.3)式和 B_t 的定义，有

$$P(S'(t) \in B_t) = \frac{|B'_t|}{n} = \sum_{i=1}^{q} \frac{m'_i(H_i)}{n} = \frac{1}{n} \sum_{i=1}^{q} m_i(H_i) \frac{f(H_i)}{\bar{f}_t} \geqslant$$

$$\frac{1}{n} \sum_{i=1}^{q} m_i(H_i) = \frac{|B_t|}{n} = P(S(t) \in B_t) \tag{10.3.10}$$

这一结果表明，集合 B_t 中的点的概率密度逐渐增加.

§10.4 隐并行性

由 10.1 和 10.2 节的讨论可知，遗传算法实质上是对模式的运算. 一个长度为 l 的串，其中隐含着 2^l 个模式. 若群体规模为 n，则其中隐含的模式个数介于 2^l 和 $n \cdot 2^l - (n-1)$ 之间. 显然，由于交叉操作的作用，并非所有的模式都能高概率地处理，因为一些具有较长的定义长度的模式将适到破坏. 本节讨论遗传算法能够有效处理的模式个数的下界.

Holland 和 Goldberg 指出，遗传算法有效处理的模式个数为 $O(n^3)$. 这意味着，尽管遗传算法每一代只对 n 个个体进行运算，但却隐含地处理了 $O(n^3)$ 个模式. Holland 命名这一性质为隐并行性(implicit parallelism). 下面来分析、推导这一结论.

设群体由 n 个长度为 l 的串构成. 只考虑那些生存概率大于 P_s(P_s 为一常数)的模式,即在单点交叉和低概率变异情况下,那些丢失概率为 $P_d<1-P_s$ 的模式. 为此,考虑那些定义长度 $l_s<P_d(l-1)+1$ 的模式.

为便于说明,不妨看一个例子. 假设要计算定义长度小于 $l_s=5$ 的隐含在下面长度 $l=10$ 的串 1011100010 中的模式.

首先,计算前 5 个位置的模式数,即计算<u>10111</u>00010 的模式数. 由于第 5 位是固定的,所以实际上是计算下列模式% % % % 1 * * * * * 的个数,其中"*"为无关符,"%"既可以代表确定值(0 或 1),也可以代表"*". 因为有 $l_s-1=4$ 个位置可以被确定或取无关符,所以这样的模式数为 $2^{l_s-1}=16$. 为了计算整个串中定义长度小于 $l_s=5$ 这类模式,将上面串的底线向右平移一位,即 1 <u>01110</u>0010,依次将串的底线向右平移 $l-l_s+1=10-5+1=6$ 次,即完成了对于这类模式数的计算.

一般地,对于长度为 l 的串,计算定义长度小于 l_s 的模式数,需将上述底线向右平移 $l-l_s+1$ 次. 由此得出,对于一个长度为 l 的串,定义长度小于 l_s 的模式数为 $2^{l_s-1}\cdot(l-l_s+1)$. 若群体规模为 n,则此类模式总数为 $n\cdot 2^{l_s-1}\cdot(l-l_s+1)$. 显然,这个结论在群体规模较大的情况下存在着对于低阶模式的重复记数的问题.

为了使模式数的估计更准确,取群体规模 $n=2^{l_s/2}$. 按照这种选取方式,希望对于阶数高于或等于$\frac{l_s}{2}$的模式最多只计数一次. 考虑到模式数目的分布呈二项式分布,因此阶数高于$\frac{l_s}{2}$的模式与低于$\frac{l_s}{2}$的模式的数目大致相等,各占一半. 如果只考虑高阶的部分,则模式数的下界为

$$n_s \geqslant n(l-l_s+1)\cdot 2^{l_s-2} \tag{10.4.1}$$

考虑到已将群体规模限制为 $n=2^{l_s/2}$,于是有

$$n_s \geqslant (l-l_s+1)\cdot\frac{n^3}{4} \tag{10.4.2}$$

即有 $n_s=cn^3=O(n^3)$,其中 c 为常数. 由此得出结论:遗传算法有效处理的模式总数正比于群体规模的立方,即其阶为 n^3.

由以上推导可见,尽管具有高阶、长定义长度的模式在交叉和变异算子作用下遭到破坏,遗传算法在处理相对小数目的串时,仍然隐含地处理了大量的模式.

§10.5 积木块假说

由 10.2 节的模式定理可知,具有低阶、短定义长度和平均适应度高于群体平均适应度的模式在子代中将以指数级增长. 由于这类模式在遗传算法中起着非常重要的作用,所以 Holland 给了它们一个特别的名称——积木块(building block).

定义 10.6 具有低阶、短定义长度和高平均适应度的模式称为积木块.

假设 10.1 (积木块假设 building block hypothesis):低阶、短定义长度、高平均适应度的模式(积木块),在遗传算于作用下,相互结合,能生成高阶、长定义长度、高平均适应度的模式,并可能最终生成全局最优解.

10.2 节的模式定理保证了较优的模式(遗传算法的较优解)的样本数呈指数级增长,从而

满足了寻找最优解的必要条件，即遗传算法存在着搜寻到最优解的可能性. 本节的积木块假设则指出，遗传算法具备寻找到全局最优解的能力，即积木块在遗传算子作用下相互拼搭、结合，能生成高阶、长定义长度、高平均适应度的模式，最终生成全局最优解.

但是，令人遗憾的是上述结论并没有得到证明，它之所以被称为假设而非定理正缘于此. 目前已有大量的实践证据支持这一假设，大量的遗传算法应用事例都表明，积木块假设在许多领域都获得了成功，例如平滑单峰问题、带干扰多峰问题以及组合优化问题等. 尽管大量的实验证据并不等于理论证明，但至少可以确信，对于我们通常碰到的许多问题，遗传算法都是适用的和有效的.

第十一章 遗传算法的实现技术

遗传算法的实现涉及到第九章所述的五个要素，即参数编码，初始群体的设计，适应度函数的设计，遗传操作设计和控制参数的设计. 每个要素对应不同的环境，存在各种相应的设计策略和方法，而不同的策略和方法决定了相应的遗传算法具有不同的性能或特征. 本章着重介绍传统遗传算法的常规实现技术.

§11.1 编 码

遗传算法主要是通过遗传操作对群体中具有某种结构形式的个体施行结构重组处理，从而不断搜索出群体中个体之间的结构相似性，形成、优化并组合积木块，以逐渐逼近最优解. 由此可见，遗传算法不能直接处理问题空间的参数，必须把这些参数转换成遗传空间的由基因按一定结构组成的染色体或个体，这一转换操作就称为编码.

一般来说，由于遗传算法的鲁棒性，它对编码的要求并不苛刻. 实际上，大多数问题都可以采用前两章所见过的基因呈一维排列的染色体表现形式，尤其是基于{0,1}符号集的二值编码形式. 但是需要指出，编码的策略或方法对于遗传操作，尤其是对于交叉操作的功能有很大影响. 在很多情况下，编码形式也就决定了交叉操作，编码策略和交叉策略是互为依存的. 因此，作为遗传算法流程中第一步的编码技术，是遗传算法理论与应用研究中需要认真考虑的课题.

首先简要介绍问题空间和 GA 空间的概念. 问题空间是指由 GA 表现型个体(有效的候选解)集所组成的空间，GA 空间是指由基因型个体(染色体)集所组成的空间.

定义 11.1 由问题空间向 GA 空间的映射(即由表现型向基因型的映射)称为编码(coding)，而由 GA 空间向问题空间的逆映射(即由基因型向表现型的逆映射)称为译码(decoding).

如果编码(包括译码)和交叉处理不当，在 GA 空间中因交叉而生成的个体逆映射到问题空间时，有可能成为无用解. 称形成无用解的染色体为致死基因. 为避免致死基因的出现，编码可以发挥一定的作用.

下面概述编码评估规范和编码原理. 评估编码策略常采用以下三个规范：

(1)完备性(completeness)：问题空间中的所有点(候选解)都能作为 GA 空间中的点(染色体)表现.

(2)健全性(soundness)：GA 空间中的染色体能对应所有问题空间中的候选解.

(3)非冗余性(nonredundancy)：染色体和候选解一一对应.

这三个评估规范是独立于问题领域的普遍准则. 对于某个具体的应用领域来说，应该客观地比较和评估该问题领域中所用的编码方法. 应该指出，严格满足上述规范的编码方法和提高遗传算法的效率并没有直接的必然的关系. 在有些问题中，允许生成致死基因的编码，尽管这样会导致冗余的搜索，但总的计算量可能反而减少，从而可以更有效地获得最优解.

上述三个编码评估规范虽然带有普遍意义，但缺乏具体的指导思想，可操作性较差. De Jong 提出了称之为编码原理的较为客观明确的编码评估准则. 由于其可操作性较好，常称之

为编码规则. 它包括如下两条规则：

(1)有意义积木块编码规则：所定编码应当易于生成与所求问题相关的短定义长度和低阶的积木块.

(2)最小字符集编码规则：所定编码应采用最小字符集以使问题得到自然的表示或描述.

规则(1)是基于模式定理和积木块假设提出的. 从应用的角度看，把握和应用与问题空间相对应的积木块的编码结构是十分困难的. 尽管如此，在有些问题的编码设计中还是应该检查一下码串中相关位(bit)位置之间的距离.

规则(2)提供了一种更为实用的编码原则. 遗传算法中通常采用二值编码设计决非偶然，因为二值编码符合规则(2)的设计思想.

应该指出，上述两个编码规则也仅仅是为编码设计提供了一定的准则. 在实际应用这些准则时仍然要针对具体问题，设计出具体有效的编码.

下面介绍几种常用的编码技术.

一、一维染色体编码

一维染色体编码是指问题空间的参数映射到 GA 空间后，其相应的基因呈一维排列构成染色体. 一维染色体编码中最常用的符号集是二值符号集{0,1}，基于此符号集的个体呈二值码串.

二值码串是目前遗传算法中常用的编码方法，它具有以下优点：

(1)简单易行；

(2)符合最小字符集编码规则；

(3)便于用模式定理进行分析，因为模式定理就是以二值编码为基础的.

二值编码的不足之处有：

(1)有一定的映射误差；

(2)不能直接反映出所求问题的本身结构特征，难以满足生成有意义积木块的编码规则.

目前，关于一维编码已有一些新的提案和方法为人们所采用，如实数表示、格雷码表示和表表示等.

二、多参数映射编码

在优化问题求解中经常会碰到多参数优化问题. 对这类问题的遗传算法常采用多参数编码. 其基本思想是把每个参数先进行二值编码得到于串，再把这些子串连成一个完整的染色体. 一般来说，多参数映射编码中的每一个于串对应各自的编码参数，因此根据各参数不同的数量级，相应的各于串可有不同的串长度.

三、可变染色体的长度编码

在自然界生物进化过程中，染色体的长度不是固定不变的. 为了融人这种机制，Goldberg 等人提出了一种称为 Messy GA(MGA)的编码方法.

MGA 主要有以下几个特点：

(1)染色体长度可变；

(2)允许过指定和欠指定；

(3)基于切断和拼接操作的交叉处理.

下面简单介绍 MGA 编码方法. 表 11.1 给出了描述 MGA 编码的一个例于.

表 11.1　MGA 编码举例

正确指定	$\{(a_1)(b_4)(c_3)\}$
过指定	$\{(a_1)(b_4)(c_3)(a_3)\}$
欠指定	$\{(a_1)(c_3)\}$

在 MGA 中，个体仍然由基因座和相应的基因构成，但基因的意义与基因座无关，而是由成对的符号所决定．个体以表的形式表示．以表 11.1 中的个体$\{(a_1)(b_4)(c_3)\}$为例，它可以解释为三位长的串，a 值为 1，b 值为 4，c 值为 3.

与标准 GA 不同，MGA 不一定要把所有的信息都用基因表现出来，因此染色体的长度或表示个体的表的长度是可变的．设表示的基本字符集为$\{a,b,c\}$，染色体长度取为 3，则正确的个体表示或编码对应表 11.1 中的正确的指定形式．染色体长度为 4 和 2 的表示分别对应表 11.1 中过指定和欠指定形式.

MGA 在做适应度计算时，需要把以表的形式表示的个体转换成规定长度的字符串的形式．对于过指定的表示，要把多余的指定除去；对于欠指定的表示，要用竞争模板来补足欠缺．因此在 MGA 中，经适应度评估后，群体中染色体的长度都相等．但经过 MGA 特有的交叉操作后，新的群体中的染色体的长度又可能不等长了．交叉操作的例子可如图 11.1 所示．在该例中，两个父本个体的长度为 9，但经过切断和拼接操作后形成分别为 6 和 12 的两个子代个体.

父代1　111|111111　→　子代1　　111|000

父代2　000000|000　→　子代2　000000|111111

图 11.1　切断和拼接操作

MGA 的可变长度染色体编码在有些应用中表现了很好的效果，但上一章所述模式定理已不再适用.

§11.2　群体设定

遗传操作是对若干个体同时进行的，这些个体组成了群体．在遗传算法处理流程中，编码设计后的第一项工作就是初始群体的设定．然后以此为起点，逐代执行遗传操作，直至满足预先设定的某种进化终止准则，获得最后群体.

群体设定的主要问题是群体规模，即群体中包括的个体数目的设定．作为遗传算法的控制参数之一，群体规模和交叉概率、变异概率等参数一样，直接影响遗传算法效能的发挥.

群体规模影响遗传优化的最终结果以及遗传算法的执行效率．当群体规模 n 太小时，遗传算法的优化性能一般不会太好．这种情况下，会使遗传算法的搜索空间中解的分布范围受到限制，因此搜索有可能停止在未成熟阶段，引起未成熟收敛(premature convergence)现象．较大的群体规模可以保持群体的多样性，避免未成熟收敛现象，以减少遗传算法陷入局部最优解的机会．但较大的群体规模意味着较高的计算复杂度．从计算效率出发考虑，群体规模越大，其适应度评估次数增加越多，所以导致计算量增加，影响算法效能．在实际应用中应当综合考虑这两个因素，选择适当的群体规模．通常 n 的取值范围为几十至几百，在优化问题中一般取 n 在 10～160 之间.

关于初始群体的设定，一般采用如下策略：

(1)根据对问题的了解，设法把握最优解所占空间在整个问题空间中的分布范围，然后，在此范围内设定初始群体.

(2)先随机生成一定数目的个体，然后从中挑出最好的个体加到初始群体中. 这种过程不断迭代，直到初始群体中个体数达到了预先确定的规模.

在遗传进化过程中，群体规模一般都维持相同大小. De Jong 在函数优化的实验中，提出了一个描述群体之间关系的参数 G，称为代沟(generation gap). 通过 G 的引用，可形成重叠群体：

$$G = \begin{cases} 1, & \text{非重叠群体} \\ 0 < G < 1, & \text{重叠群体} \end{cases}$$

在重叠群体模式下，每次从群体中选取 nG 个个体参与遗传操作，而其他 $n(1-G)$ 个个体直接保存到下一代群体中. 代沟方式的选用为遗传算法利用优化过程的历史信息提供了条件，加速了遗传算法的收敛过程. 当代沟 G 过小时可能导致遗传算法的过早的不成熟收敛. 一般取 G 在 0.3～1 之间.

§11.3　适应度函数

遗传算法在进化搜索中仅以适应度函数为依据，而不需要外部信息. 遗传算法的适应度函数不受连续可微的限制，且其定义域可以为任意集合. 对适应度函数的惟一要求是，针对输入能够计算出可加以比较的非负结果. 这一特点使得遗传算法具有较广的应用范围.

适应度函数评估是选择操作的依据，适应度函数的设计直接影响到遗传算法的性能. 在具体应用中，适应度函数的设计要视求解问题本身的要求而定. 本节主要介绍适应度函数设计的基本准则和要点.

一、目标函数映射成适应度函数

在优化问题求解中，有些问题的目标是求费用函数(代价函数)$g(x)$的最小值，有些问题的目标是求效能函数(或利润函数)$g(x)$的最大值. 即使某一问题可自然地表示成求最大值的形式，但也不能保证对于所有的 x，$g(x)$都取非负值. 由于在遗传算法中要根据适应度函数值计算选择概率，所以要求适应度函数的值要取非负值.

当求解问题本身是最大化问题时，为了保证适应度函数的非负性，可采用如下变换式：

$$f(x) = \begin{cases} g(x) + C_{\min}, & \text{当 } g(x) + C_{\min} > 0 \\ 0, & \text{其他情况} \end{cases}$$

式中系数 $C_{\min}$ 可以是一个合适的输入值，或是当前群体或前 k 代群体中 $g(x)$的最小值.

当求解问题本身是最小化问题时，为了保证适应度函数的非负性，可采用如下变换式：

$$f(x) = \begin{cases} C_{\max} - g(x), & \text{当 } g(x) < C_{\max} \\ 0, & \text{其他情况} \end{cases}$$

式中系数 $C_{\max}$ 可以是一个合适的输入值，或是当前群体或前 k 代群体中 $g(x)$的最大值.

二、适应度函数定标(scaling)

在遗传算法中，群体中个体被选择参与竞争的个数与适应度有直接关系. 为此，有必要调整群体中各成员的竞争水平，以获得适应度最高的个体.

在遗传进化初期，通常会出现一些超常个体. 若按比例选择策略，则这些超常个体有可能在群体中占很大比例，从而可能导致未成熟收敛现象. 这些异常个体因竞争力太突出会控制选择过程，影响算法的全局优化性能. 此时，应设法降低这些异常个体的竞争能力，这可以通过缩小相应的适应度函数值来实现. 另外，在遗传进化过程中，虽然群体中个体多样性尚存在，但往往会出现群体的平均适应度已接近最佳个体适应度，在这种情况下，个体间竞争力减弱，最佳个体和其他大多数个体在选择过程中有几乎相等的选择机会，从而使有目标的优化过程趋于无目标的随机漫游过程. 对于这种情形，应设法提高个体间竞争力，这可以通过放大相应的适应度函数值来实现. 这种对适应度的缩放调整称为适应度定标.

定标方式主要有以下几种：

1. 线性定标

设原适应度函数为 f，定标后的适应度函数为 f'，则线性定标可采用下式表示：

$$f' = af + b \tag{11.3.1}$$

其中系数 a、b 的选择需满足：

①定标后适应度平均值 $\bar{f}'$ 与原适应度平均值相等.

②定标后适应度函数最大值 f'_{max} 等于原适应度平均值 $\bar{f}$ 的所指定的倍数，即

$$f'_{max} = C\bar{f} \tag{11.3.2}$$

其中 C 是最优个体期望达到的复制数. 实验表明，对于不大的群体，如 n 介于 50～100 之间，取 C 在 1.2～2 之内，已经获得了成功的应用.

上述选取(11.3.1)式中系数 a、b 的条件①的提出，是为了保证在后继的选择操作中，具有平均适应度值的个体可贡献一个期望的子孙到下一代. 条件②的提出是为了控制原适应度值最大的个体可贡献子孙的数目.

值得注意的是，在使用线性定标时有可能出现负适应度值的情况. 这主要是在算法运行的后期，对原适应度函数进行了过分的缩放. 图 11.2 所示为正常线性定标情形，此时少数异常个体的适应度被缩小，同时，为数不多的个体适应度被扩大，所以定标后的适应度未出现负值. 而图 11.3 情况则不同，由图中可见，在一些个体的适应度值远小于群体的平均适应度值 $\bar{f}$ 和最

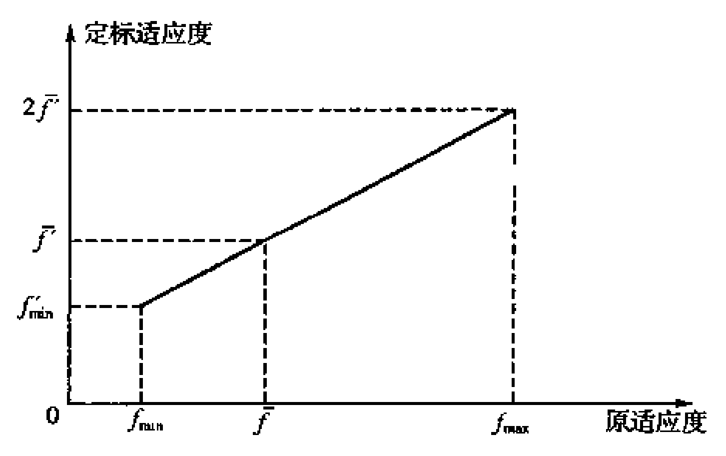


图 11.2　正常线性定标

大适应度值 f_{max}，且 $\bar{f}$ 和 f_{max} 又很接近的情形，为了提高群体中个体的竞争力，以避免随机漫游现象而采用线性定标. 希望把比较接近的 f_{max} 和 $\bar{f}$ 拉开时，原来低适应度值经定标后就会变为负值. 解决此类问题的办法很多，当不能将系数 C 调整到期望值时，可以简单地把原适应度

最小值 f_{min} 映射到定标后适应度最小值 f'_{min}. 但此时仍要保持 $\bar{f}'=\bar{f}$.

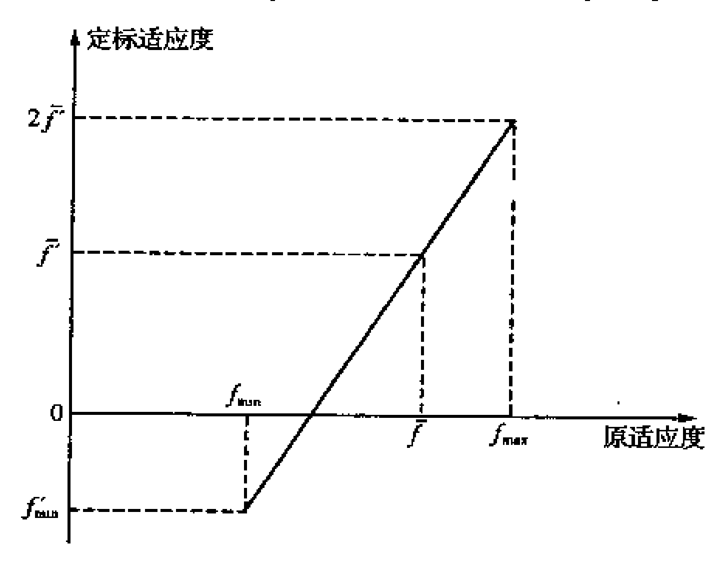


图 11.3　线性定标出现负适应度值的情况

综上所述，线性定标公式(11.3.1)中的系数 a、b 可根据两点式确定，具体方法如下：

首先，利用

$$\begin{cases}\bar{f}'=\bar{f}\\ f'_{max}=C\bar{f}\end{cases}$$

确定

$$\begin{cases}a=\dfrac{C-1}{f_{max}-\bar{f}}\bar{f}\\ b=\dfrac{f_{max}-C\bar{f}}{f_{max}-\bar{f}}\bar{f}\end{cases}$$

然后，对定标后适应度的非负性进行判别，若 $f'_{min}>0$，即

$$f_{min}>\frac{C\bar{f}-f_{max}}{C-1}$$

则采用上述关于 a、b 的选择. 否则，利用

$$\begin{cases}\bar{f}'=\bar{f}\\ f'_{min}=0\end{cases}$$

确定

$$\begin{cases}a=\dfrac{\bar{f}}{\bar{f}-f_{min}}\\ b=-\dfrac{f_{min}\bar{f}}{\bar{f}-f_{min}}\end{cases}$$

2. 幂函数定标

该定标方式定义如下：

$$f'=f^k \tag{11.3.3}$$

其中幂指数 k 为一常量，与求解问题有关，需要在理论分析估计的基础上，结合实验进行调整才能获得最好的结果.

3. 指数定标

这种定标方式定义为

$$f' = e^{-kf} \tag{11.3.4}$$

其中 k 为常量. 这种机制是受模拟退火方法的启发得到的.

下面应用指数定标方法处理两组典型的数据.

例 1　设群体中有 6 个串,其中一个串的适应度值特别大:

原适应度值	200	8	7	6	5	4
定标适应度值($k=-0.005$)	2.718	1.041	1.036	1.030	1.025	1.020

例 2　设群体中有 6 个串,它们的适应度值都比较接近:

原适应度值	9	8	7	6	5	4
定标适应度值($k=-0.5$)	90	55	33	20	12	7

从上面的例子可见,指数定标既可以让非常好的串保持多的选择机会,同时又限制了其选择数目,以免其很快控制整个群体. 这种方法也提高了相近串间的竞争性. 系数 k 的值是非常重要的,它决定了选择的强制性,k 越小,选择强制性就越趋向于那些具有最高适应度值的串.

选取合适的适应度定标方案是非常重要的. 采用这一技术的好处在于:第一,可以使遗传算法的选择强度通过适应度函数值的放、缩加以控制. 一般来说,在遗传算法的初始阶段,希望选择强度稍低一些,以免遗传群体被单个或少数几个适应度较高的个体所支配. 而在遗传算法的后期,即算法接近收敛的情况下,由于群体内的个体间适应度的差异较小,继续优化的潜能较低,因此有必要适当地提高选择强度,以便遗传算法收敛到更优的解. 第二,实际应用中可能会出现目标函数值小于零的情况,此时若直接使用目标函数作为适应度会引起随机选择方面的问题,因此利用适应度函数定标技术可以将目标函数值映射到非负实数值域. 第三,使用适应度定标技术还可处理某些无目标函数值的组合优化问题. 此时可以根据群体内个体性能的相对优劣,以一定的函数映射方法对个体性能进行适应度定标,并依此计算选择概率.

三、适应度函数与问题约束条件

在实际应用中,许多优化问题都是带约束条件的. 由于遗传算法仅靠适应度来评估和引导搜索,所以求解问题所固有的约束条件不能明确地表示出来. 用遗传算法求解此类问题需要考虑一些对策.

可以采用一些很自然的方法来处理约束条件,即在进化过程中,每迭代一次就检测一下新的个体是否违背了约束条件. 如果没有违背,则作为有效个体保留,反之,作为无效个体从群体中除去. 这种处理方法对于弱约束问题求解还是有效的,但对于强约束问题求解则难于奏效,因为此时寻找一个有效个体的难度可能不亚于寻找最优个体.

作为一种对策,可采用在目标函数中引入惩罚函数的方法. 该方法的基本思想是设法对个体违背约束条件的情况给予惩罚,并将此惩罚体现在适应度函数设计中. 这样,一个约束优化问题就转换为一个附带考虑惩罚项的无约束优化问题.

设要解决的约束最小化问题可描述为

$$\begin{cases} \min f(\boldsymbol{x}) \\ \text{st } h_i(\boldsymbol{x}) = 0, (i = 1, 2, \cdots, m) \\ g_i(\boldsymbol{x}) \geqslant 0, (i = m+1, \cdots, p) \end{cases} \tag{11.3.5}$$

其中 $\boldsymbol{x}$ 为一向量.

通过惩罚函数法，上述问题可转化为无约束优化问题：

$$\min f(x)+\sum_{i=1}^{m}r_iH[h_i(x)]+\sum_{i=m+1}^{p}r_iG[g_i(x)] \tag{11.3.6}$$

其中 H 和 G 为惩罚函数，r_i 为惩罚系数.

惩罚函数可以有多种确定方法，如可以取为

$$\begin{cases}H[h_i(x)]=h_i^2(x)\\G[g_i(x)]=\mathrm{e}^{-g_i(x)}\end{cases} \tag{11.3.7}$$

在一定的条件下，当惩罚系数 r_i 的取值接近无穷大时，无约束解可收敛到约束解.

把惩罚加到适应度函数中的思想是简单而直观的. 但是，惩罚函数值在约束边界处会发生急剧的变化，常常会有较大的问题，要加以注意. 用遗传算法求解约束问题的对策除了从适应度函数设计着手外，还可以在编码设计和遗传操作设计等方面采取一定的措施.

§11.4　遗传操作

在遗传算法中，通过编码生成初始群体后，接下来的任务就是对群体的个体按照它们对环境的适应程度(适应度评估)进行遗传操作，以实现优胜劣汰的进化过程. 遗传操作可使问题的解逐代优化，并逼近最优解.

遗传操作包括以下三个基本遗传算子：选择(selection)、交叉(crossover)和变异(mutation). 这三个遗传算子有如下特点：

(1)它们的操作都是在随机扰动情况下进行的，即遗传操作是随机化操作. 因此，群体中个体向最优解迁移的规则是随机的. 但是需要指出，这种随机化操作和传统的随机搜索方法是有区别的. 遗传操作进行的是高效有向的搜索，而不同于一般随机搜索方法所进行的无向搜索.

(2)遗传操作的效果除了与编码方法、群体规模、初始群体以及适应度函数的设定有关外，还与上述三个遗传算子所取的操作概率密切相关.

(3)三个遗传算子的操作方法随具体求解问题的不同而异，它们和个体的编码方式直接相关.

由于二值编码是最常用的编码方法，所以，对遗传算子的论述以二值编码为基础.

11.4.1　选择算子

从群体中选择优胜的个体，淘汰劣质的个体的操作称为选择. 选择算子亦称为再生算子(reproduction operator). 选择操作建立在对群体中个体的适应度进行评估的基础上，目前常用的选择方法有如下几种：

一、*适应度比例方法*(fitness proportional model)

适应度比例方法是目前遗传算法中最基本也是最常用的选择方法，也称为赌轮选择(roulette wheel selection)或蒙特卡罗选择(Monte Carlo Selection). 在这种选择机制中，个体每次被选中的概率与其在群体环境中的相对适应度成正比.

设群体规模为 n，其中第 i 个个体的适应度值为 f_i，则其被选择的概率为

$$P_{si}=\frac{f_i}{\sum_{j=1}^{n}f_j} \tag{11.4.1}$$

选择概率 P_s 反映了第 i 个个体的适应度在整个群体的个体适应度总和中所占的比例. 个体适应度值越大,其被选择的概率就越高,反之亦然.

现以表 11.2 为例,将赌轮选择方式的计算机实现步骤说明如下:

表 11.2　赌轮选择示例

个体序号	1	2	3	4	5	6	7	8	9	10
适应度	8	2	17	7	2	12	11	7	3	7
适应度累计值	8	10	27	34	36	48	59	66	69	76
随机数	23	49	76	13	1	27	57			
被选中的个体号	3	7	10	3	1	3	7			

表 11.2 描述了赌轮选择方式. 表中第一行说明有 10 个个体参与选择,第二行表示各个体的适应度,第三行标记适应度的累计值,总值为 76. 选择操作执行时,首先在[0,76]区间内产生均匀分布的随机数,如第四行所示. 然后依次序将第三行的累计适应度与随机数相比较,其值大于或等于随机数的第一个个体列为入选的选择对象. 例如,第一个随机数是 23,除了 1 号、2 号个体外,其余个体的累计适应度均大于 23,然而 3 号个体累计值为 27,是第一个大于 23 的个体,所以它入选.

上述选择过程,可描述如下:

(1)依次累计群体内各个体的适应度,得相应的累计值 S_i,最后一个累计值为 S_n;

(2)在[0,S_n]区间内产生均匀分布的随机数 R;

(3)依次用 S_i 与 R 相比较,第一个出现 S_i 大于或等于 R 的个体 i 被选为选择对象;

(4)重复(2),(3)直至满足所需要的个体数目.

表面上看,个体的选择是随机的. 但实际上选择是依据相邻两个适应度累计值的差值

$$\Delta S_i = S_i - S_{i-1} = f_i$$

进行的,式中 f_i 为第 i 个个体的适应度. 事实上,适应度 f_i 越大,ΔS_i 的距离越大,随机数落在这个区间的可能性越大,第 i 个个体被选中的机会也越多. 图 11.4 描述了这种情形. 图中的指针固定不动,外圈的圆环可以任意转动,圆环中每段对应于适应度的大小. 从统计意义讲,适应度越大的个体被选中的机会越大. 适应度小的个体尽管被选中的概率小,但仍有可能被选中,这样就增加了个体的多样性,便于执行交叉及变异. 所以赌轮选择方法既体现了"适者生存"的原则,又保持了个体性态的多样性.

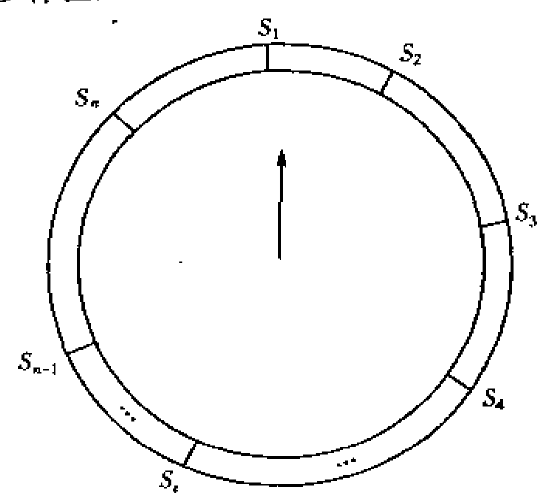


图 11.4　赌轮选择原理

二、最佳个体保留方法(elitist model)

该方法首先按适应度比例选择方法执行遗传算法的选择操作，然后将当前解群体中适应度最高的个体直接复制到下一代群体中.它的主要优点是能够保证遗传算法终止时得到的结果一定是历代出现过的具有最高适应度的个体.但是，这也隐含了一种危机，即局部最优个体的遗传基因会急剧增加而使进化有可能陷于局部最优解.

三、期望值方法(expected value model)

在执行赌轮选择机制时，若群体规模较小，则依据产生的随机数有可能会不正确地反映个体适应度的选择，即适应度高的个体可能被淘汰，而适应度低的个体可能被选择.为了克服这种随机误差，可以采用期望值方法.

期望值方法采用如下步骤：首先计算群体中每个个体在下一代生存的期望数目.

$$R_i = \frac{f_i}{\bar{f}} = \frac{nf_i}{\sum_{j=1}^{n} f_i} \tag{11.4.2}$$

然后按期望值 R_i 的整数部分安排个体被选中的次数.而对期望值 R_i 的小数部分，可按确定方式或随机方式进行处理.确定方式是将 R_i 的小数部分按值的大小排列，自大到小依次选择，直到被选择个体数达到群体规模为止.随机方式可按赌轮选择机制进行，直到选满为止.

四、排序选择方法(rank-based model)

该方法首先根据适应度大小顺序对群体中的个体排序，然后把事先设计好的概率表按序分配给个体，作为各自的选择概率.选择概率和适应度无直接关系，而仅与个体排列序号有关.这种方法的优点是，可以较好地处理比例选择机制存在的两个问题，即个别或极少数适应度值相当高的染色体的基因在群体中迅速遗传而导致的群体不成熟收敛，以及当群体内个体适应度彼此相互接近时导致的纯粹随机选择.该方法的缺点是选择标准过于偏离个体的适应度.

五、随机联赛选择方法(stochastic tournament selection model)

设定联赛规模 M，从群体中任意选择 M 个个体，将其中适应度最高的个体保存到下一代.这一过程反复进行，直到选择进入下一代的个体数达到群体规模为止.联赛规模 M 通常取为 2.

六、排挤方法(crowding model)

在重叠群体模式(代沟 $0<G<1$)下，采用排挤方法可以使新生成的子代排挤相似的旧父代个体，以提高群体的多样性.

该方法执行步骤如下：首先设定排挤系数 CF(crowding factor)，然后从群体中随机地挑选 CF 个个体组成个体集(新的个体不包括在内)，从该个体集中淘汰一个与下一代群体中新个体的海明距离最短的个体，将获胜的个体直接保存到下一代群体中.

以上介绍的是常用的几种选择方法，在具体使用时，应根据求解问题的特点选用合适的方法，或将不同的选择机制混合运用，以期在整体上改善算法的优化性能.

11.4.2　交叉算子

遗传算法中起核心作用的是遗传操作的交叉算子.交叉是指对两个父代个体的部分结构进行重组而生成新个体的操作.交叉算子的设计应与编码设计协调进行，使之满足交叉算子的评估准则，即交叉算子需保证前一代中优秀个体的性状能在下一代的新个体中尽可能得到遗传和继承.

对二值编码来说，交叉算子包括两个基本内容：一是从由选择操作形成的配对库(mating pool)中，对个体随机配对，并按预先设定的交叉概率 P_c 决定每对是否需要交叉操作. 二是设定配对个体的交叉点(cross site)并对配对个体在这些交叉点前后的部分结构进行交换.

下面针对二值编码介绍几种基本的交叉算子.

一、单点交叉(one—point crossover)

具体操作是，在个体串中随机设定一个交叉点，实行交叉时，对两个配对个体在该点前后的部分结构进行互换，生成两个新个体.

下面给出一个单点交叉的例子.

个体 A　　0011|11000→001100111　　新个体 A′

个体 B　　1010|00111→101011000　　新个体 B′

在本例中，交叉点设置在第 4 和第 5 个基因座之间. 交叉时，该交叉点后的两个个体的部分码串互相交换. 于是，个体 A 的第一到第四个基因与个体 B 第五到第九个基因组成一个新的个体 A′. 同理，可得到个体 B′. 交叉点是随机设定的，若染色体长为 l，则可能有 $l-1$ 个交叉点设置.

二、二点交叉(two—point crossover)

首先随机设定两个交叉点，实行交叉时，对两个配对个体在这两个交叉点之间的码串进行互换，生成两个新个体.

一个二点交叉的例子如下：

个体 A　　0011|110|00→001100100　　新个体 A′

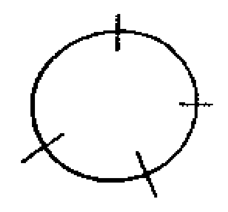
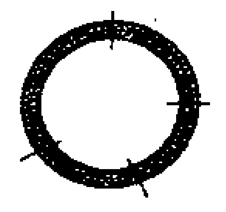
个体 B　　1010|001|11→101011011　　新个体 B′

在本例中，二个交叉点分别设置在第四基因座和第五基因座以及第七基因座和第八基因座之间. A，B 两个个体在这两个交叉点之间的码串相互交换，生成新个体 A′和 B′. 若个体长为 l，则对于二点交叉来说，可能有 $\frac{1}{2}(l-1)(l-2)$ 种交叉点的设置.

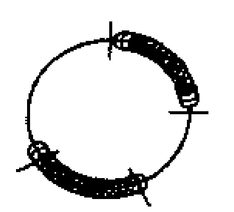
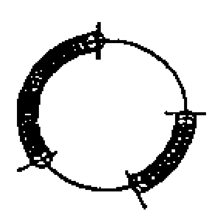
三、多点交叉(multi—point crossover)

多点交叉是上述两种交叉的推广. 若用参数 CP 表示交叉点数，则当 $CP=1$ 时，即为单点交叉. 当 CP 为偶数时，根据交叉互换规律，染色体可以当做基因环来处理. 如 CP 等于 4 时，交叉情况如图 11.5 所示. 当 CP 为奇数时，它可以等效为偶数点交叉. 此时可假定在码串头部(即基因座 0)有一个交叉点. 图 11.6 给出了 $CP=3$ 时的交叉情况.

若染色体长为 l，则多点交叉共有 C_{l-1}^{CP} 种交叉设置可供随机挑选. 当 CP 较大时，每种交叉点设置被随机选中的可能性很小，因此能被保存的部分结构也就很少. 由于多点交叉不能有效地保存重要模式，所以较少为人们采用.

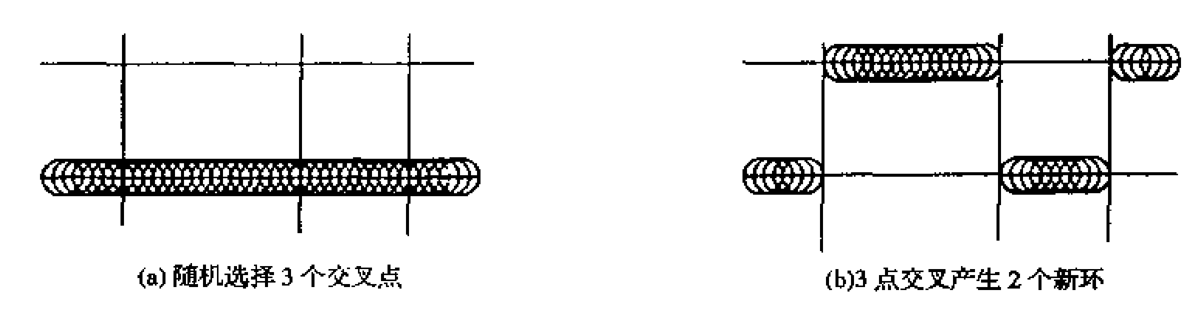



(a) 随机选择 4 个交叉点

(b) 4 点交叉产生 2 个新环

图 11.5　偶数点交叉



(a)随机选择 3 个交叉点　　(b)3 点交叉产生 2 个新环

图 11.6 奇数点交叉

四、一致交叉(uniform crossover)

一致交叉是通过设定屏蔽字(mask)来决定新个体的基因继承两个旧个体中哪个个体的对应基因. 当屏蔽字中的某位为 1,则交叉该位所对应的父本的基因,否则,不交换. 下面给出一个一致交叉的例子.

旧体体 A	111000
旧体体 B	000111
屏蔽字	010101
新个体 A′	101101
新个体 B′	010010

11.4.3 变异算子

变异算子的作用是改变群体中个体串的某些基因座上的基因值. 对于由字符集{0,1}生成的二值码串来说,变异操作就是把基因座上的基因值取反,即 1→0 或 0→1.

变异算子操作的步骤为:首先在群体中所有个体的码串范围内随机地确定基因座,然后按预先设定的变异概率 P_m 对这些基因座的基因值进行变异.

遗传算法引入变异的目的,一是使算法具有局部随机搜索的能力,二是维持群体多样性. 当遗传算法通过交叉算子的作用已接近最优解邻域时,利用变异算子的局部随机搜索能力可以加速向最优解收敛. 显然,此时变异概率应取较小值,否则接近最优解的积木块会因变异而遭到破坏.

在遗传算法中,交叉算子因其全局搜索能力而作为主要算子,变异算子因其局部搜索能力而作为辅助算子. 遗传算法通过交叉和变异这一对既相互配合又相互竞争的操作而使其具备兼顾全局和局部的均衡搜索能力. 当群体在进化中陷于搜索空间中某个超平面面仅靠交叉不能摆脱时,通过变异操作可有助于这种摆脱. 而当通过交叉操作,算法已形成所期望的积木块时,变异操作又有可能破坏这些积木块. 如何有效地配合使用交叉和变异操作,是提高遗传算法效能的一个重要研究课题.

下面介绍几个常用的变异算子.

一、基本变异算子

执行基本变异操作时,首先在个体码串中随机挑选一个或多个基因座,然后以概率 P_m 对这些基因座的基因值做变动.

下面给出{0,1}二值码串中基本变异操作的一个例子.

$$\text{个体 A}\quad 10\,\underline{0}10\,\underline{1}10 \xrightarrow{\text{变异}} 10110010 \quad \text{个体 A}'$$

二、逆转算子(inversion operator)

逆转算子的基本操作是，在个体码串中随机挑选两个逆转点，然后将两个逆转点间的基因值以概率 P_i 逆向排序. {0,1}二值码串的逆转操作举例如下：

$$\text{个体 A}\qquad 10\underbrace{11010}_{\text{逆转点}}00 \xrightarrow{\text{逆转}} 100101100 \qquad \text{个体 A}'$$

由此例可见，通过逆转操作，个体 A 中从基因座 3 到基因座 7 之间的基因排列得到逆转，即从序列 11010 变成 01011.

逆转操作可以等效为一种变异操作，但它的真正目的在于实现一种重新排序(reordering). 这里所谓重新排序是指对个体的基因排列进行重新组合，但又不影响整个个体的特征(适应度值)的一种操作. 在自然界生物的基因重组中就有这种重新排序的机制. 在遗传算法中采用这种重新排列的目的是为了提高积木块的繁殖率. 在用遗传算法求解某些问题时，群体中有些个体的基因排列常常会出现如下情况，即有助于形成积木块的某些基因相隔较远，此时采用一般的交叉操作会破坏相应的积木块的生成. 在这种情况下就有必要对这样的个体进行重新排列操作.

为了利用逆转算子来实现这种重新排序功能，必须把基因值的意义与基因座的位置独立开来，以保证经过重新排序的个体其适应度不变. 为此，可采用个体扩展表示法，现举例说明如下：

$$\text{个体 A}\quad \begin{matrix}123456789\\ 10\underbrace{11010}_{\text{逆转点}}00\end{matrix} \xrightarrow{\text{逆转}} \begin{matrix}127654389\\ 100101100\end{matrix}\quad \text{个体 A}'$$

在本例中，每个基因都用整数 1～9 命名或编号. 这些命名或编号记述了各个基因与基因座的关系，即各个基因译码的含义. 例如，基因 6 的译码含义是 8(二-十进制译码). 经过逆转操作后，基因 6 虽然被移动，但它的译码含义因仅和其命名或编号有关，而与基因座位置无关，所以它的译码含义仍为 8(而不是 32). 这样对经过扩展表示的个体 A 施行逆转操作后，生成的个体 A 的基因被重新排列，但其适应度值仍然保持与原个体一致.

三、自适应变异算子(adaptive mutation operator)

自适应变异算子与基本变异算子的执行步骤完全一致，不同之处在于基本变异操作过程中变异概率 P_m 始终保持不变，而在执行该算子过程中，P_m 随群体中个体的多样性程度的改变或其他指标的变化而自适应调整. 这些参照指标可以根据具体问题设定，例如，可以根据交叉操作所得两个新个体的海明距离作为参照指标，海明距离越小 P_m 越大，反之 P_m 越小.

第十二章　遗传算法的若干改进研究

§12.1　避免陷于局部极小的遗传算法

为了避免遗传算法在搜索的过程中陷入局部最优解问题，本节给出了一种新的方法，在选择父染色体时不仅要考虑染色体的适应度，而且还要根据搜索结果动态地调整各染色体被选中的概率，并以易于陷入局部极小的多峰函数为例进行验证，模拟计算结果表明这一方法对克服遗传算法收敛于局部根小是十分有效的.

一、陷入局部极小的原因

在GA中染色体的优劣是用适应度来评价的，一般地，适应度越大的染色体所代表的解可能越优，其生存的概率也越大. 从概率上说，适应度高的染色体产生优化的子染色体的概率大，使GA收敛于全局极小点的概率也大. 因而，传统的GA是以群体中各染色体的适应度为依据进行选择操作的，即适应度大的染色体被选作父本的概率也大. 各染色体被选中作为父本的概率可用下式给出：

$$P_i = \frac{f(C_i)}{\sum_{i=1}^{n} f(C_i)} \tag{12.1.1}$$

其中 n 为群体中染色体数.

如果在搜索过程中经过若干代(generation)的进化仍然找不到最优解，这说明在以前各代中仅以适应度为依据所选取的父染色体并不能产生最优的个体. 而传统GA的选择操作仍以适应度为依据来选择父染色体，不断的再生，致使群体中不断地加入和父染色体相近的个体，它们的适应度不断地接近平均适应度，使得这些非最优个体或模式在群体中所占的比例不断加大，占据了统治地位，进而产生了遗传漂移现象，最后GA陷入局部极小，即收敛于局部最优解.

二、改进的遗传算法

在选择父染色体时除考虑染色体的适应度外，还依据经过若干代再生的结果动态调整选择概率，改进的GA算法步骤如下：

(1) 定义调节系数 α 和阈值 θ，其中 α 和 θ 为小正数；设定调节周期为 K 代；设定群体中染色体数 N；适应度函数 $f(\quad)$.

(2) 首先对随机产生的初始群体完成下列操作：

①求初始群体中各染色体的适应度，并将

$$\max_{i=1}^{N}(f(i)) \rightarrow f_1$$

②将初始群体中所有染色体备份组成一个保留群体，以便陷入局部根小时恢复.

③用(12.1.1)式求出的概率 P_i 来选择父染色体，并对当选为父的染色体进行备份.

(3) 初始化再生代计数器 $I=0$.

按传统的 GA 实现遗传操作，并对当选为父染色体的染色体进行备份，若找到最优解则算法结束；否则 $I = I + 1$.

(4) 取最大适应度值赋给 f_2，

$$\max_{i=1}^{N}(f(i)) \rightarrow f_2$$

若

$$f_2 - f_1 \geqslant \theta$$

则说明搜索过程仍朝优化方向进展，用当前群体中的染色体替换保留群体中的染色体，并将 $f_2 \rightarrow f_1$，转(3)；

若

$$f_2 - f_1 < \theta \text{且} I = K$$

则说明收敛过程进展缓慢，可能已陷入局部极小，此时用保留群体中的染色体替换当前群体中的染色体以使群体恢复到 K 代以前的状态，并将在前 K 代再生中曾被选作父本的染色体以及与父本染色体海明距离相近的染色体的选择概率降低，即

$$P_i = \alpha P_i$$

转(3).

(5) 转(4).

三、实验结果

所用的例子是多峰值函数的优化，共进行了三组实验，所选用的函数为：

$$y_1 = (10 - 0.2x)\sin(0.8\pi x)$$

$$y_2 = (10 - 0.2x)\sin(0.4\pi x)$$

$$y_3 = (10 - 0.2x)\sin(0.2\pi x)$$

它们都是峰值不同的多峰函数，都易于过早地陷入局部极小. 由于各函数的周期不同，过早地陷入局部极小的程度也不一样. 下面给出染色体的编码和参数的设定.

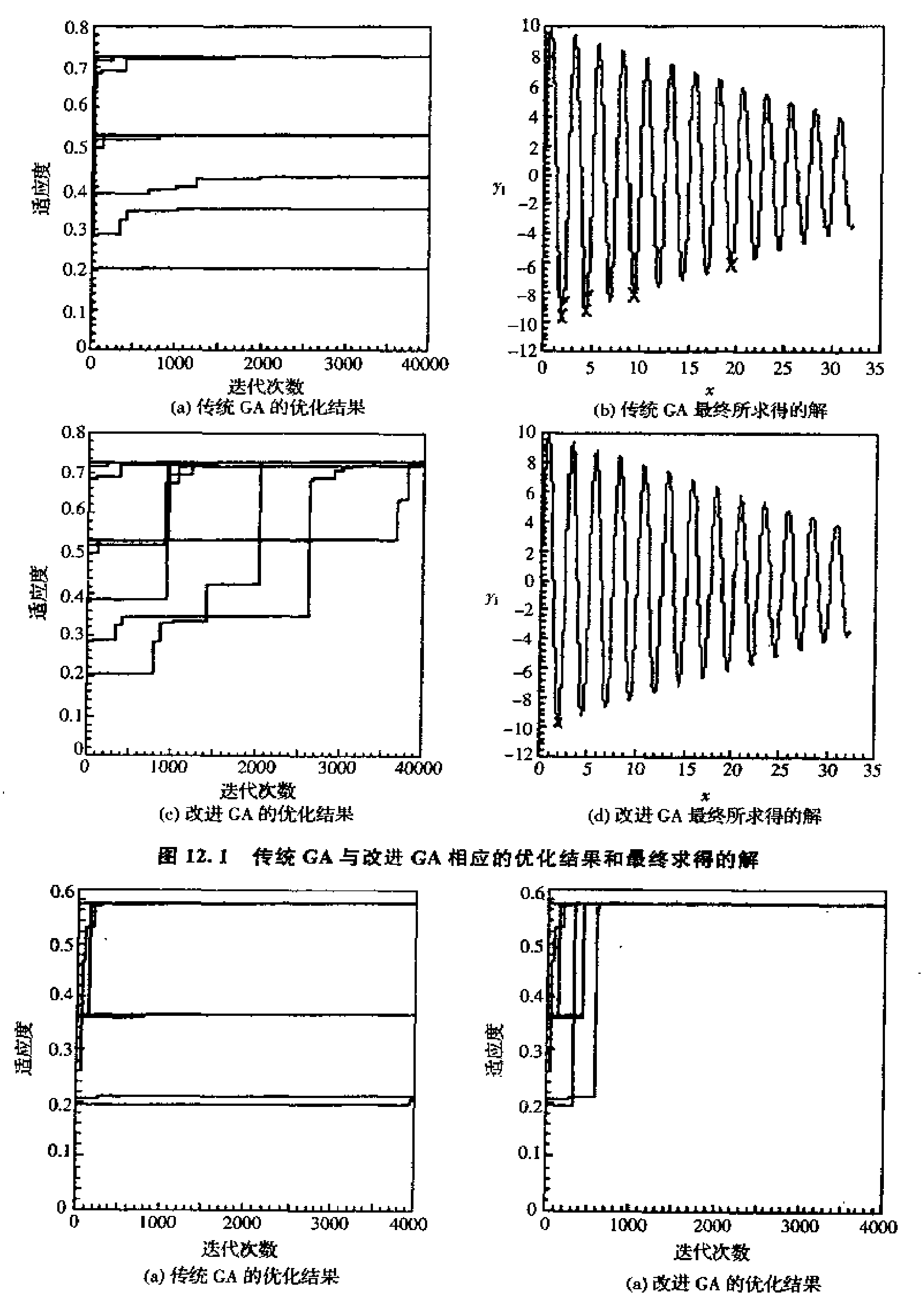
所选用的函数定义域为[0,32]，染色体的编码为{0,1}上的位串，长度为 15，代表一个二进制数. 从左起前 5 位表示整数部分，后 10 位表示小数部分，例如：

$$(100101100010000)_2 = (10010.1100010000)_2 = (18.765625)_{10}$$

参数的选取为：群体中染色体数 N 为 20，变异率为 0.01，调节系数 α 为 0.1，阈值 θ 为 0.05，调节周期 K 为 300，适应度函数定义为 $f = \dfrac{1}{y+11}$.

对于每个函数，选取不同种类的随机数模拟 10 次. 图 12.1 为 y_1 的优化结果，其中图 12.1(a)为传统 GA 的优化结果，图 12.1(c)为改进后 GA 的优化结果，图 12.1(b)为传统 GA 最终所求得的解，图 12.1(d)为改进后 GA 所得的解. 图 12.2 和图 12.3 分别为对函数 y_2 和 y_3 的模拟结果.

从实验结果可以看出改进后的 GA 能有效地防止陷入局部极小，在多数情况下都能找到全局极小点或全局准极小点，而在相同的参数下，传统 GA 却易于陷入局部极小点. 本算法的不足之处是计算量较大，与传统 GA 相比收敛速度有所降低.



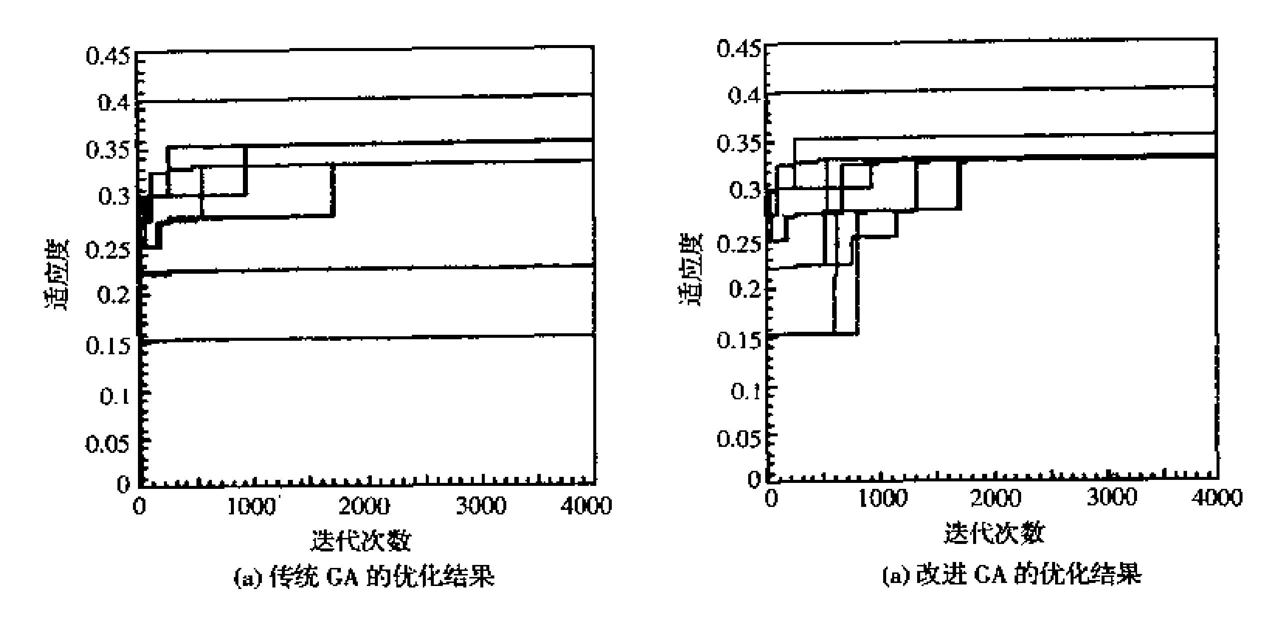
(a) 传统 GA 的优化结果　(b) 传统 GA 最终所求得的解

(c) 改进 GA 的优化结果　(d) 改进 GA 最终所求得的解

图 12.1　传统 GA 与改进 GA 相应的优化结果和最终求得的解

(a) 传统 GA 的优化结果　(a) 改进 GA 的优化结果

图 12.2　传统 GA 与改进 GA 相应的优化结果



(a) 传统 GA 的优化结果　　(a) 改进 GA 的优化结果

图 12.3　传统 GA 与改进 GA 相应的优化结果

§12.2　一种快速收敛的速传算法

GA 比起其他搜索算法，主要优点是简单、通用、鲁棒性强，但其不足之处是由于搜索空间大，计算量大，因而收敛速度慢. 特别是，当求解的问题复杂时，使得所构造的染色体的结构及长度增大，搜索空间增大，收敛速度慢的问题就越显得严重. 本节针对这一问题，提出了一种改进的快速收敛的遗传算法，并以八皇后问题为例加以验证，实验结果表明了这一方法的有效性.

一、快速收敛的遗传算法

传统的遗传算法是依据求解问题来确定染色体编码形式及定义串长度，然后初始化随机生成的 N 个染色体并组成一个群体，再对群体中的染色体实施遗传操作以搜索代表优化解的染色体.

快速收敛遗传算法增加了对染色体的分割(dividing)与重组(recombination)操作. 分割意指将染色体的定长串分割成若干段，每段内可能包含若干个基因. 该算法首先依据于各段的结构和段长，分别初始化随机生成的 N 个染色体段(chromosome segment)组成的段群体(segmented population)，然后对各段群体独自实施遗传操作以寻找优化段，最后，再对选出的优化段实施重组操作，重新组合成完整的染色体来搜索优化解.

快速收敛遗传算法的步骤如下：

(1) 确定染色体 C 的编码形式，选定 C 的适应度函数 $F(C)$.

(2) 对染色体 C 按其结构及长度 L 实施分割操作，生成 k 个染色体段. 基于每个染色体段各自随机地生成 k 个段群体，$PU_i=\{C_{ij}\}$，其中 $i(i=1,2,\cdots,K)$ 为段群体号，$j(j=1,2,\cdots,N)$ 为段群体内染色体号.

(3) 定义染色体段的段适应度函数为：

$$f^{g+1}(C_{ij}^{g+1}) = F(C_j^{g+1}) = F(C_{11}^{g+1} \cup C_{21}^{g+1} \cup \cdots \cup C_{i-1,1}^{g+1} \cup C_{ij}^{g} \cup C_{i+1,1}^{g} \cup \cdots \cup C_{k1}^{g}) \tag{12.2.1}$$

其中上标 $g(g\geqslant 1)$ 表示代数，$C_{i1}^{g}(i=1,2,\cdots,k)$ 是依据段适应度函数 $f_g(C_{ij}^{g})$ $(j=1,2,\cdots,N)$

求得的第 g 代的各段群体中的最优染色体段，而 $g+1$ 代的各段群体中的各染色体段的适应度值由 $f^{g+1}(C_{ij}^{g+1})$ 求得，对第 i 段群体求其 $g+1$ 代最优染色体段 C_{i1}^{g+1} 时要用到 $g+1$ 代最优染色体段（当段号小于 i 时）和 g 代最优染色体段（当段号大于 i 时），依据(12.2.1)式求得的 $f^{g+1}(C_{ij}^{g+1})$ 越大，表示段 C_{ij}^{g+1} 越优，最后在 N 个染色体段中选取一个最优的命名为 C_{i1}^{g+1}. 由(12.2.1)式定义的染色体的段适应度函数可见，f^{g+1} 是动态变化的，即 f^{g+1} 随 g 代和 $g+1$ 代各段群体中的最优染色体段的不同而不同.

(4) 依据(12.2.1)式计算各 PU_i 中染色体的段适应度 $f^{g+1}(C_{ij}^{g+1})$，并将段群体内的染色体分别按其段适应度值由大到小排序.

(5) 搜索过程以下面循环体实现.

```
g=0;
for( ; ;){
        g=g+1;
        for (i=1;i<=k;i++)
           { 对段群体 PU_i 按传统的遗传算法实施遗传操作产生 2m 个子染色体,2m<<N ;
           替换 PU_i 中段适应度值小的 2m 个染色体,即 PU_i 中的后 2m 个染色体;
           for (j=1;j<=N;j++)
              {C_1^{g+1}=C_{11}^{g+1}∪C_{21}^{g+1}∪…∪C_{i-1,1}^{g+1}∪C_{i,j}^g∪C_{i+1,1}^g∪…∪C_{k1}^g ;
              计算 F(C_1^{g+1});
              if(C_1^{g+1} 满足目标解)
                     解码 C_1^{g+1},得到目标解,算法结束;
              else
                        f^{g+1}(C_{ij}^{g+1})=F(C_1^{g+1});
              }
           按 PU_i 中段适应度值由大到小将染色体重新排序;
           }
        }
```

假定一次遗传操作产生 $2m$ 个子染色体，那么传统的 GA 需要计算适应度的次数为 $2m$ 次；改进的算法对于一个段群体的一次遗传操作若产生 $2m$ 个于染色体，则实际上对于 K 个段群体来说相当于产生了 $2mN^{k-1}k$ 个子染色体，而对于适应度的计算量为 KN 次；对于传统的 GA 来说，若产生 $2mN^{k-1}k$ 个于染色体，则计算子染色体适应度的量亦为 $2mN^{k-1}k$ 次，显然下式成立：$2mN^{k-1}k > kN$. 因而改进的算法在产生同样个数的子染色体时能大大减少其适应度的计算量，所以能大幅度加快收敛过程.

二、实验结果

本节以八皇后问题为例对快速收敛的遗传算法进行了验证. 八皇后问题是一个 NP 问题，是把八个皇后摆在 8×8 的棋盘上，使得没有一个皇后能俘获任何别的皇后，即每一行、列仅有一个皇后，每条对角线上至多有一个皇后.

1. 染色体的编码

八皇后问题的解可以用一个 8×8 矩阵来表示：

$$A=\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{18} \\ & \vdots & & \\ A_{41} & A_{42} & \cdots & A_{48} \\ & \vdots & & \\ A_{81} & A_{82} & \cdots & A_{88} \end{bmatrix} \tag{12.2.2}$$

染色体的编码为{0,1}上长度为 64 的位串,1 表示该位置有皇后,0 表示该位置无皇后.

2. 染色体的分割

本实验将染色体分别分割为 2 段、4 段和 8 段,下面给出当 $K=4$ 时各段所包含的位串.

$$\begin{aligned} &(A_{11}\cdots A_{18}A_{21}\cdots A_{28})\in PU_1 \\ &(A_{31}\cdots A_{38}A_{41}\cdots A_{48})\in PU_2 \\ &(A_{51}\cdots A_{58}A_{61}\cdots A_{68})\in PU_3 \\ &(A_{71}\cdots A_{78}A_{81}\cdots A_{88})\in PU_4 \end{aligned} \tag{12.2.3}$$

3. 适应度函数的选取

定义函数:

$$E_1=\sum_{i=1}^{8}\left|\sum_{j=1}^{8}A_{ij}-1\right| \tag{12.2.4}$$

$$E_2=\sum_{j=1}^{8}\left|\sum_{i=1}^{8}A_{ij}-1\right| \tag{12.2.5}$$

$$E_3=\sum_{k=1}^{2*8-1}\delta\Big(\sum_{i=1}^{8}\sum_{j=1}^{8}g_1(A_{ij})-1\Big),\qquad g_1(A_{ij})=\begin{cases}A_{ij}, & \text{若 } i+j-1=k \\ 0, & \text{否则}\end{cases} \tag{12.2.6}$$

$$E_4=\sum_{k=1}^{2*8-1}\delta\Big(\sum_{i=1}^{8}\sum_{j=1}^{8}g_2(A_{ij})-1\Big),\qquad g_2(A_{ij})=\begin{cases}A_{ij}, & \text{若 } i-j+8=k \\ 0, & \text{否则}\end{cases} \tag{12.2.7}$$

$$E=E_1+E_2+E_3+E_4 \tag{12.2.8}$$

其中 i,j 分别表示行号和列号;

$$\delta(x)=\begin{cases}x, & \text{若 } x>0 \\ 0, & \text{若 } x\leqslant 0\end{cases} \tag{12.2.9}$$

上述诸函数的定义使得 $E_1=0$ 和 $E_2=0$ 保证每行和每列上仅有一个皇后,$E_3=0$ 和 $E_4=0$ 保证每条对角线上至多有一个皇后,显然满足目标解时 $E=0$. 适应度函数定义为:

$$F=\frac{1}{E+1} \tag{12.2.10}$$

由定义可知 $0<F\leqslant 1$. 当找到目标解时 $F=1$,算法结束.

图 12.4 为使用传统遗传算法解决八皇后问题的适应度变化曲线,图 12.5 为使用快速收敛的遗传算法时的适应度变化曲线,其中图 12.5 给出了 $k=2,4,8$ 的三种情况. 从图 12.4 和图 12.5 对比可以看出快速收敛的遗传算法极大加快了收敛速度. 另外,使用该方法对函数如 $\lg x$ 和 $\sin x$ 的求解也进行了模拟实验,收到了同样的效果.

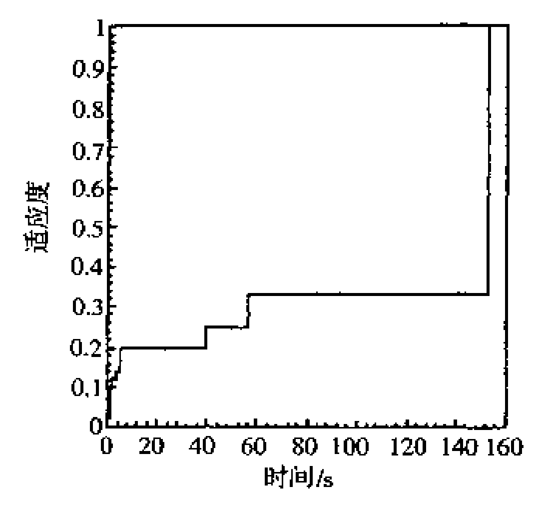


图 12.4　采用传统 GA 的适应度变化曲线

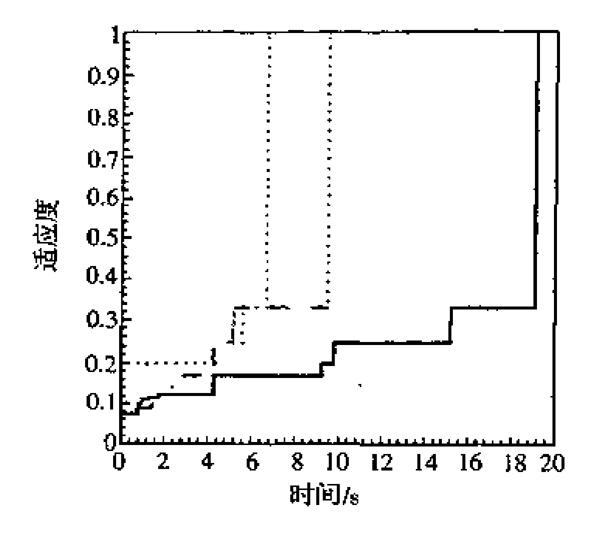


图 12.5　采用快速 GA 的适应度变化曲线

§12.3　求解多值优化问题的回溯遗传算法

对于许多问题，合理的解决方案不止一个，在很多情况下，需要得到所有的或者大部分目标解，这时 GA 便无能为力了. 原因是 GA 在选择父本时是以适应度为依据的，最优解一经产生，它的适应度就要高于其他所有的染色体，因此会引导全局的搜索继续朝这个方向进行，而将种群中其他信息逐渐抛弃，最后导致种群中充满了与这种模式相同或相近的染色体，这种状态一经形成便很难再产生其他的最优解. Deb K 等人较系统地研究了多峰函数的优化问题. 所采用的方法是用群集来保持种群的多样性，但这种方法在很多情况下还是不够的. 针对这类问题本节提出了一种回溯的遗传算法(Backtracking Genetic Algorithm, BGA)，这种方法的基本思想是：算法在迭代过程中每找到一最优解，种群便恢复到若干代前的状态，并对种群中被标记做过父本的及那些与父本海明距离相近的染色体的适应度值进行自适应调节使之降低，从而在其后的搜索过程中降低其被选作父本的概率，以引导搜索朝其他的最优解方向进行，使之避免产生相同的解. 本节以一多峰函数 $Y=\sin(X)$ 和图着色问题为例对 BGA 的有效性进行了说明.

一、回溯的遗传算法(BGA)

1. BGA 的特点

BGA 较 GA 相比有以下特点：

(1) BGA 中染色体的适应度值可在搜索过程中自适应地进行调节.

(2) 设置一堆栈，算法每迭代若干次，便将种群中所有的染色体作为一集合压栈保存以便回溯时恢复. 与普通堆栈不同的是，在把栈顶的染色体集合恢复为当前种群时，该集合依然保存在栈中.

(3) 算法每迭代一次都把栈顶集合中的全部或部分染色体与所选的父本相比较，如果其中某条与所选的父本相同或在海明距离上相近，则对栈顶集合中的该染色体进行标记.

(4) 对于最优解适应度值事先可确定的问题，例如本节中的图着色问题，如果找到一个最优解或者经历若干代不再收敛，算法回溯. 对于最优解适应度值事先不可确定的问题，例如 TSP 问题，如果算法经历若干代不再收敛，则表明算法可能已找到了最优解，算法回溯. 算法在每次回溯之前都把当前最好的染色体的编码及其适应度值记录下来.

2. 回溯机制

定义 12.1　m 为一参数，如果一染色体集合中被标记做过父本或与父本海明距离相近（海明距离确定视情况而定）的染色体总数小于 m，则该集合为有效集合(valid set).

定义 12.2　如果一非栈底染色体集合中被标记做过父本或与父本海明距离相近（海明距离确定视情况而定）的染色体总数 $\geq m$，则该集合为无效集合(invalid set).

定义 12.3　当算法回溯将一染色体集合恢复为当前种群时，根据该集合中每条染色体做父本的标记状况将那些被标记过的染色体的适应度值乘以一参数 $\alpha(0<\alpha<1)$ 使之降低，称 α 为调节系数(adjusting factor).

定义 12.4　当算法迭代过程中，如果经过 BF 代仍不能产生更好的子染色体，则算法回溯. BF 为一参数，称之为回溯因子(backtracking factor).

回溯的方法是：

① 检查栈顶所保存的染色体集合是否为有效集合，如果是无效集合则废弃该集合，将其从栈中清除并返回 ① .

② 若为有效集合，则根据该集合中每条染色体做父本的标记状况，将那些被标记过的染色体的适应度值乘以调节系数 α 使之降低，并将集合中所有染色体重新排序. 废弃当前的种群，恢复栈顶的染色体集合为当前种群，并连同各染色体调节后的适应度值一同恢复. 栈顶的染色体集合仍保存在栈中. 这里，栈底的集合永远都是有效集合.

因为 GA 再生时选择父本是以适应度为依据的，将那些做过父本或与父本相近的染色体的适应度降低，使在以后的再生过程中降低其被选做父本的概率，以避免生成相同的子染色体，从而引导搜索朝其他的最优点进行.

BGA 的其他操作如交叉、变异均与 GA 相同.

二、算法步骤

在介绍算法前先介绍一下算法所用的主要的常量、变量的作用.

常量：SIZE，BF，β，N，n，I_{max}，α.

变量：g，i，s.

BF 为回溯因子. 算法每隔 SIZE 代将种群中所有染色体为一集合压栈保存，SIZE 称为步长. N 为种群中染色体总数. 每次再生选择 n 对父本进行交叉、变异产生 $2n$ 条子染色体. α 为调节系数. β 为父本选择系数. 变量 g 记录算法总的迭代次数. 变量 i 为从一最近的回溯点算起算法的迭代次数. s 记录解的序号. I_{max} 为算法最大迭代次数.

在实验中由于采用的例子中染色体的适应度函数公式只与染色体编码有关，和迭代次数及其他因素无关，所以每次迭代只计算新生成的染色体适应度.

对子最优解适应度值确定的问题算法如下：

```
main(    )
    {初始化. 随机生成 N 条染色体为初始种群，求其适应度值，并按适应度值由高到低
    的顺序排序；g=0；i=0；s=1；
    for (  ;  ;  )
        { if ( g>I_max)
            break;
        if (g % SIZE = 0)
```

```
            将当前种群为一集合压栈保存.
            依据当前种群中各染色体排序状况选择 n 对父本，进行交叉、变异产生 2n 条
            子染色体，求其适应度值，并与种群中原有的 N 条染色体一起重新排序，如果
            一新生子染色体与某条原有的染色体适应度值相同，那么，新生的子染色体将
            排在前面，淘汰后 2n 条染色体.
            将栈顶集合中各条染色体依次与这一次再生所选择的父本相比较，若相同或
            距离相近(标准视情况面定)则对其进行标记.
            g=g+1;
            i=i+1;
            if (产生目标解且与前些次迭代所产生的目标解不同)
                {记录该解;
                s=s+1;i=0;
                算法回溯;
                }
        else if (i≥ BF )
                { i=0;
                算法回溯;
                }
        }
    }
```

对于最优点适应度值不确定的问题算法如下：

```
main(      )
        {初始化. 随机生成 N 条染色体为初始种群，求其适应度值，并按适应度值由高
        到低的顺序;g=0;i=0;s=1;
        for ( ; ; )
        { if ( g>I_max)
          break;
        if ( g % SIZE = 0 )
          将当前种群为一集合压栈保存.
          依据当前种群中各染色体排序状况选择 n 对父本，进行交叉、变异产生 2n 条
          子染色体，求其适应度值，并与种群中原有的 N 条染色体重新排序，如果一新
          生子染色体与某条原有的染色体适应度值相同，那么，新生的子染色体将排在
          前面，淘汰后 2n 条染色体.
          将栈顶集合中各条染色体依次与这一次再生所选择的父本相比较，若相同或
          距离相近(标准视情况而定)则对其进行标记.
          g=g+1;
          i=i+1;
          if ( i ≥ BF )
            { 记录该解;
```

```
            s=s+1;
            i=0;
            算法回溯;
            }
        }
    }
```

以上是本节提出的回溯的遗传算法.该算法可以有效地处理多解问题,但此算法有一不足之处就是在每次选择父本时,都要将其与栈顶集合中的每条染色体相比较,开销大.因此,对这种算法做了进一步改进,算法与前者大致相同,只是在计算父本与栈顶集合中的染色体海明距离时并不与栈顶集合中所有的染色体相比较,而只是与前若干条(D 条)染色体相比较.这是因为 GA 是以适应度值为依据来选择父本的,一条染色体的适应度值越大,其被选做父本的概率越大,而种群中的染色体始终保持按适应度值由高到低的次序排序的,因此,种群中染色体序列越往前其被选做父本的概率越大,这样的比较越有意义.反之,越靠后的染色体被选做父本的概率越小,这种比较也就失去了意义.因此,只要参数选择适当,这种方法能在迭代次数基本不增多的情况下有效地减小开销.

根据每条染色体在种群中的排序状况,参照父本选择系数 β 选出 n 对父本,并将其复制,其中 $0<\beta<1, n\leqslant N$,每个父本选择的方法如下:

① $i\leftarrow 1$;

② 随机产生一个[0,1]区间的小数 σ,若 $\sigma\leqslant\beta$,则选择第 i 条染色体作为父本.结束;

③ $i\leftarrow i+1$,若 $i\leqslant N$,返回②.

④ 当 $i\geqslant N$,选择第一条染色体作为父本,结束.

由此可见,适应度值最高的染色体被选择的概率为 $\beta(1-\beta)^{N-1}$,适应度值次高的染色体被选择的概率为 $\beta(1-\beta)$,再次是 $\beta(1-\beta)^2$,第 $i(i>1)$ 条染色体被选择的概率为:

$$P(i)=\beta(1-\beta)^{i-1} \tag{12.3.1}$$

依此类推,直到最后一条染色体被选做父本的概率为 $P(N)=\beta(1-\beta)^{N-1}$.若所选一对父本相同,则废弃其中一条,并按上述方法重新选择,直至两父本不同.D 值的选取和父本的选择系数 β 有直接关系.种群中第 i 条染色体被选做父本的概率如(12.3.1)式,由此式可得出:

$$\sum_{j=i+1}^{N}P(j)=(1-\beta)^{i}-(1-\beta)^{N} \tag{12.3.2}$$

经验得出:一般来说,如果 $\sum\limits_{j=i+1}^{N}P(j)<5\%$,则表明从第 $i+1$ 条染色体起一直到最后一条被选做父本的概率很小,这时取 $D\geqslant i$,对算法的迭代次数不会有太大影响.因此,D 应满足:

$$(1-\beta)^{D}<5\%+(1-\beta)^{N} \tag{12.3.3}$$

三、实验结果

1.问题说明

实验中所取的多峰函数为 $Y=\sin(X)$,如图 12.6 所示,定义域在[0,32]之间,存在 5 个极小点,本实验是要利用 BGA 找出所有极小点的 X 值或其近似值.

图着色问题是著名的易于描述难于解决的 NP 完全问题.实验中所用的例于是一个 10 点四色问题.如图 12.7 所示,有(1,2,…,10)个点,两点间有连线则代表两点相邻,否则代表两点

不相邻.有红、黄、蓝、绿四种颜色,任何相邻两点颜色都不同的着色方案则是一合法的配色方案.对于该问题整个搜索空间有 $4^{10}=1048576$ 个点,有 24 种合法的着色方案.本实验的目的是要利用 BGA 找出所有的或大部分合法的着色方案,而这对于 GA 来讲是不可能的.

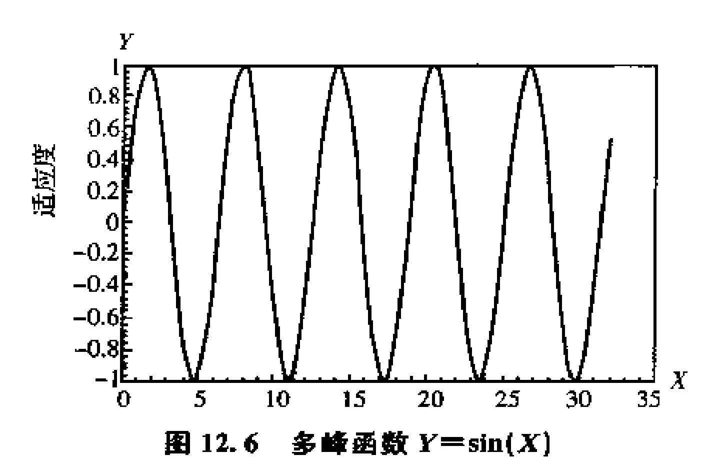


图 12.6　多峰函数 $Y=\sin(X)$

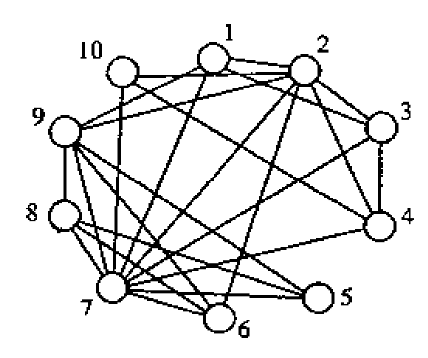


图 12.7　着色问题示意图

2. 染色体编码

对于多峰函数 $Y=\sin(X)$ 的优化问题,染色体编码为定义在字母表{0,1}上的字符串,长度为 15 位.每条染色体代表一个[0,32]区间的一个正数,前面 5 位代表整数部分,后 10 位代表小数部分.如 $(000111010100000)=(00011.1010100000)_2=(3.65625)_{10}$.

对于四色问题,染色体编码为定义在字母表{0,1,2,3}上的长度为 10 的定长串.0 代表红色,1 代表黄色,2 代表蓝色,3 代表绿色.由左到右的每一位分别代表由 1,…,10 的各个点的颜色.整个一条染色体则代表一种着色方案.

3. 适应度函数的确定

对于多峰函数 $Y=\sin(X)$ 的优化,适应度函数确定为:

$$f_1(C)=\frac{1}{\sin X+2} \tag{12.3.4}$$

显然 $f_1\leqslant 1$. f_1 越接近 1,表明此染色体所代表的解越接近最优解.

对于四色问题,用一 10×10 矩阵来描述各点的邻接情况,如(12.3.5)式所示,

$$\boldsymbol{A}=\begin{bmatrix}0&1&1&0&0&0&1&0&1&0\\1&0&1&1&0&1&1&0&1&1\\1&1&0&1&0&0&1&0&0&0\\0&1&1&0&0&0&1&0&0&1\\0&0&0&0&0&0&1&1&1&0\\0&1&0&0&0&0&1&1&1&0\\1&1&1&1&1&1&0&1&1&1\\0&0&0&0&1&1&1&0&1&0\\1&1&0&0&1&1&1&1&0&0\\0&1&0&1&0&0&1&0&0&0\end{bmatrix} \tag{12.3.5}$$

对于任意 $A[i][j]\ (i\neq j)$,如果 $A[i][j]=1$,代表 i,j 两点相邻.如果 $A[i][j]=0$,则代表 i,j 两点不相邻,对于任何 $i=j(1\leqslant i\leqslant 10)$,$A[i][j]=0$;

对于一染色体 $C=(g_1g_2\cdots g_{10})$,g_i 代表第 i 个基因,适应度函数确定为

$$f_2(C)=\frac{1}{\sum_{i=1}^{10}\sum_{j=1}^{10}\delta(i,j)+1} \tag{12.3.6}$$

其中

$$\delta(i,j)=\begin{cases}1, & \text{如果 } A[i][j]=1 \text{ 且 } g_i=g_j \\ 0, & \text{否则}\end{cases} \tag{12.3.7}$$

由(12.3.6)、(12.3.7)式可见当C为一合法的着色方案的编码时$f_2(C)=1$.

对于$Y=\sin(X)$的优化问题，实验中种群染色体总数N取值为50.每次迭代取$n=1$对父本进行交叉、变异产生两条子染色体.调节系数$\alpha=0.1$.父本选择系数$\beta=0.3$，步长SIZE=90，回溯因子BF=100. BGA1则每选择一对父本与栈顶集合中所有的染色体进行比较. BGA2每选择一对父本只与栈顶集合中前20条($D=20$)染色体进行比较，BGA3则每选择一对父本只与前10条($D=10$)染色体进行比较，对BGA1、BGA2、BGA3各以不同种类的随机数模拟了多次，基本上都能在2000代左右找到所有解.三者在迭代次数上无太大差别.表12.1给出了利用BGA1、BGA2、BGA3所得的优化结果中较好的一种情况.三者均只需四次回溯便找出5个最优点，说明了BGA的有效性.因算法本身的随机性的特点表中所示的迭代次数并不能完全说明BGA1、BGA2和BGA3在最终迭代次数上的差别，但可算出它们迭代一次所需的时间不同.从表中可以得出：

BGA1平均每迭代一次所需时间为29.710448/639≈0.0465 s；

BGA2平均每迭代一次所需时间为22.44108/559≈0.04015 s；

BGA3平均每迭代一次所需时间为27.041702/709≈0.0381 s.

BGA与GA相比，额外的开销主要体现在父本与栈顶集合中的染色体的比较标记上.其他操作如压栈、出栈，算法迭代若干代才做一次，所需额外时间开销很少，因此可以忽略不计.实验得出：

BGA1较GA平均每迭代一次所需的额外开销约为29%；

BGA2较GA平均每迭代一次所需的额外开销约为11.5%；

BGA3较GA平均每迭代一次所需的额外开销约为5.8%.

可见BGA2、BGA3较BGA1能够有效地减小额外开销，但D值的选取要适当.若D值过大，开销增大，而D值过小，对回溯之后的父本选择又不能正确指导.实验中β取值为0.3，则$D\geqslant 9$时，BGA2、BGA3较BGA1在最终迭代次数上相差不大，实验也说明了这一点.

对于着色问题，实验中种群的染色体总数N取值为40，每次再生选择$n=1$对父本进行交叉、变异生成两个子染色体.调节系数$\alpha=0.5$，父本选择系数$\beta=0.3$，步长SIZE=300，回溯因子BF=500. BGA1每次迭代把栈顶集合中所有染色体与这一次所选择的父本相比较，对当中与所选父本相同的或海明距离相近的染色体进行标记. BGA2只与前20条($D=20$)相比较，BGA3则只与前10条($D=10$)相比较.对BGA1、BGA2和BGA3都以不同种类的随机数模拟了多次，三者基本上都能在30000代左右找到所有解.所需迭代次数大致相同.表12.2给出了利用BGA1、BGA2、BGA3所得的优化结果中较好的一种情况.实验得出：

BGA1迭代一次所需时间为285.450213/19892≈0.01435 s；

BGA2迭代一次所需时间为268.256893/20720≈0.01295 s；

BGA3迭代一次所需时间为253.230991/20528≈0.01234 s；

BGA1 较 GA 平均每迭代一次所需的额外开销约为 23.28%；

BGA2 较 GA 平均每迭代一次所需的额外开销约为 11.28%；

BGA3 较 GA 平均每迭代一次所需的额外开销约为 5.76%.

可见 BGA2、BGA3 能较 BGA1 在不增加迭代次数的情况下显著地减小额外开销.因算法本身的随机性的影响，所以所得的结果并不完全成比例.

表 12.1 多峰函数 $Y=\sin(X)$ 的优化结果

BGA1				BGA2				BGA3			
序号	所得解	时间/s	代数	序号	所得解	时间/s	代数	序号	所得解	时间/s	代数
1	10.995	3.901	14	1	10.995	4.313	30	1	29.844	3.803	18
2	29.843	11.294	186	2	29.875	7.795	145	2	4.719	8.682	172
3	17.281	15.389	303	3	4.718	12.635	279	3	10.996	13.204	312
4	4.718	22.419	471	4	23.561	17.380	428	4	23.566	17.087	411
5	23.562	29.710	639	5	17.281	22.441	559	5	17.278	27.042	709

表 12.2 着色问题的优化结果

BGA1			BGA2			BGA3		
序号	时间/s	代数	序号	时间/s	代数	序号	时间/s	代数
1	2.734508	181	1	5.037816	396	1	5.087691	422
2	10.486272	721	2	5.589505	423	2	5.417225	449
3	11.518246	793	3	16.419802	1258	3	6.029761	502
4	11.944172	822	4	22.421091	1719	4	7.127107	591
5	13.248576	909	5	31.384891	2416	5	9.608229	800
6	15.030547	1032	6	34.762043	2657	6	13.149103	1101
7	23.352355	1618	7	38.833499	2972	7	13.573693	1136
8	30.860417	2135	8	40.07766	3067	8	14.332897	1200
9	33.783619	2345	9	46.534582	3582	9	21.046492	1707
10	38.56951	2673	10	64.688589	5034	10	29.605318	2402
11	48.207811	3353	11	67.918973	5283	11	31.274239	2527
12	50.337842	3501	12	77.285939	6031	12	37.731053	3075
13	51.596589	3585	13	97.557337	7581	13	38.72743	3157
14	52.924076	3679	14	101.853254	7910	14	56.925708	4645
15	58.919918	4089	15	111.389333	8608	15	61.136168	5002
16	60.148106	4172	16	131.378538	10189	16	87.813869	7168
17	75.313213	5235	17	144.690393	11198	17	96.075987	7844
18	110.829625	7654	18	146.922869	11374	18	115.733969	9418
19	113.787599	7856	19	152.750574	11834	19	116.872501	9507
20	140.750044	9755	20	154.101437	11937	20	136.622501	11102
21	189.347032	13191	21	178.166726	13772	21	160.569276	13108
22	213.240405	14840	22	183.45132	14176	22	191.778211	15621
23	237.87432	16564	23	267.285283	20651	23	217.780371	17739
24	285.450213	19892	24	268.256893	20720	24	253.230991	20528

第十三章　遗传算法的收敛性

目前遗传算法已经在机器学习、人工智能、自适应控制、人工神经网络训练、图像处理、复杂组合优化问题求解等方面得到了越来越广泛的应用. 人们对遗传算法已经进行了大量的改进,并使之应用于更广泛的领域. 但对于遗传算法收敛性以及收敛速度等问题的研究,与遗传算法的应用研究相比,可谓甚少. 遗传算法源于自然选择和生物遗传学,相对于其鲜明的生物基础,遗传算法的理论基础公认是不完善的. 这种不完善主要表现在缺少广泛而完整的有关遗传算法的收敛性理论. 由于遗传操作的随机性,使得人们难于将传统的遗传算法纳人恰当的便于分析的数学模型,因此使得对传统的遗传算法的收敛性分析成为十分困难的问题. 目前关于遗传算法收敛性的研究都是在对于算法增加某些限制条件后得到的遗传算法的各种修正形式的基础上进行的.

关于遗传算法收敛性理论的研究大多是将群体视为一个随机变数,利用随机过程的数学模型,通过对随机变数的概率密度分布的研究来考察随机变数的演变的. 20 世纪 90 年代中期的研究工作包括:1994 年,Qi X F 和 palmieri F 利用条件概率与边缘分布进行研究,给出了演变的公式,但不是马尔可夫(Markov)过程,没有给出演变的最终结果;Rudolph G 建立了一种遗传算法的有限齐次马尔可夫过程的数学模型,并证明了该过程不收敛到全局最优解;Yao L 和 Sethares W A 提出了一种改进的遗传算法,并将其应用于非线性参数估计,利用随机变数的均值性质分析了遗传算法的收敛性问题;1996 年,徐宗本、高勇分析了遗传算法中过早收敛的起因与特征,提出了一种可以预防和克服过早收敛的改进的遗传算法,并证明了该改进算法依概率收敛到全局最优状态;王丽微等人从知识表示与算符相结合的角度考察遗传算法,提出了一个遗传算法收敛的充分条件等等.

关于遗传算法收敛性的研究是进化计算理论研究的一个重要课题,它对于改进遗传算法以及对于遗传算法的应用研究都具有重要的理论意义和指导意义. 但是迄今为止,人们还未曾得到一个关于遗传算法的完整的收敛性分析结果. 特别地,关于各种改进的遗传算法的收敛速度估计目前尚无任何结果,这是当前最迫切需要解决的问题之一. 因为它能从理论上对遗传算法的任何修正形式提供评判标准,以指明改进遗传算法效能的正确方向.

本章首先讨论未成熟收敛及其对策,然后对标准遗传算法和另外两种改进的遗传算法的收敛性问题进行分析.

§13.1　未成熟收敛

未成熟收敛是遗传算法中不可忽视的现象,它主要表现在两个方面:

(1) 群体中所有的个体都陷于同一局部极值而停止进化;

(2) 接近最优解的个体总是被淘汰,进化过程不收敛.

导致产生遗传算法未成熟收敛的因素在算法运行的各环节都可能出现,具体表现为:

(1) 在进化初始阶段,生成了具有很高适应度的个体 X;

(2) 在基于适应度比例的选择下，大部分个体与 X 一致，其他个体被淘汰；

(3) 相同的两个个体进行交叉操作，从而不能生成新个体；

(4) 经变异操作生成的新个体的数量少，被淘汰的概率大；

(5) 群体中的大部分个体都处于与 X 一致的状态.

根据可能出现未成熟收敛的上述情况，应该在编码、适应度函数和遗传操作等设计中考虑相应的对策，以抑制或避免未成熟收敛现象. 此处列出几种可供参考的对策：

(1) 适当提高变异概率. 在进化初始阶段，它可以加强遗传算法的随机搜索能力.

(2) 调整选择概率. 例如可以把选择概率本身也作为个体进行优化.

(3) 对适应度函数进行适当定标.

(4) 维持群体中个体的多样性. 可以采用如下具体措施：① 增大群体规模，但要考虑计算量增加的因素；② 实施局部化：把群体分成若干子群体，每个子群体独立地进行选择操作，这样可使由于出现不适当个体而产生未成熟收敛的现象局部化；③ 实施单一化：控制群体中相同个体出现的个数；④ 增大配对个体距离：可利用海明距离来判断配对个体的相似程度，并依此决定是否配对.

上述策略对于抑制未成熟收敛的效果可能依待解决的优化问题的不同而异，因此需经实验检验来确定采用何种对策.

§13.2 标准遗传算法的收敛性分析

若将群体中的个体视为一个状态，则群体中的各种个体的集合可以视为一种状态的分布. 这种状态的分布根据遗传算法的运行而演变. 由于遗传算法的运行是具有随机性的，其基本操作只与当前的状态有关(与演变的历史无关)，是无后效的，因此可以把群体内的个体视为一个具有不同状态的随机变量的概率密度分布. 于是随着时间 $t(t=1,2,\cdots)$ 的推移，这个随机变数的分布在不断演变. 在一定条件下，可以把它看成一个有限的马尔可夫链(序列).

下面利用 Markov 链来分析标准遗传算法的收敛性问题.

一、标准遗传算法与 Markov 链

考虑全局优化问题

$$\max\{f(x):x\in D\subset \mathbf{R}^n\},f:D\subset\mathbf{R}^n\to\mathbf{R}^1 \tag{13.2.1}$$

遗传算法基于以下两条策略求解上述优化问题：①对于给定的目标函数 f，遗传算法使用 f 的任一值域非负并与 f 有相同根值点的适应度函数；②遗传算法不直接作用于实变量 x，而是作用于 x 的某种编码(通常为定长二进制数串编码). 所以，对于取定 f 的任一适应度函数 F 和固定长度为 L 的二进制数串编码，遗传算法通过求解组合优化问题

$$\max\{F(a):b\in E\} \tag{13.2.2}$$

来求解问题(13.2.1). 其中 $E=\{0,1\}^L$ 为 D 的编码空间，即 D 中所有实变量的长度为 L 的二进制数串编码全体.

求解问题(13.2.2)的标准遗传算法概述如下：

1. 初始化

①确定种群规模 m、杂交概率 $P_c\in[0,1]$、变异概率 $P_m\in[0,1]$及终止进化准则；

②从 E 中随机选取 m 个个体 $b_i(0)$ 组成初始种群

$$b(0)=\{b_1(0),b_2(0)\cdots,b_m(0)\};$$

③ 计算 $b_i(0)$ 的适应度 $F(b_i(0))$；

④ 置 $k=0$.

2. 种群变化

①对每一 $b_i(k)$ 依据其适应度赋一繁殖概率 $p_i(k)$；

②以概率 $p_i(k)(1\leqslant i\leqslant m)$ 从 $b(k)$ 中随机选取两个个体（如设为 $b_{i1}(k),b_{i2}(k)$）；

③依概率 p_c 对 $b_{i1}(k),b_{i2}(k)$ 进行杂交操作，产生两个中间个体 S_1',S_2'；

④依概率 P_m 对 S_1',S_2' 进行变异操作，产生两个新个体；

⑤计算由上述①～④所产生的 $2m$ 个新个体的适应度，对这 $2m$ 个个体连同 $b(k)$ 由某种选择规则确定 m 个个体组成的新一代种群

$$b(k+1)=\{b_1(k+1),b_2(k+1),\cdots,b_m(k+1)\}$$

3. 终止检验

如果 $b(k+1)$ 已满足预设的进化终止准则，则停止，并输出最优解；否则置 $k=k+1$，转步骤 2.

标准遗传算法的种群序列的分布的演变是一个以 E 为状态空间的 Markov 链. 由遗传操作所引起的种群中基因的随机改变可以由 $N\times N(N=2^{Lm})$ 阶转移概率矩阵 $\boldsymbol{P}$ 来记录，它可以分解为随机矩阵的乘积，

$$\boldsymbol{P}=\boldsymbol{C}\cdot\boldsymbol{M}\cdot\boldsymbol{S} \tag{13.2.3}$$

其中 $\boldsymbol{C},\boldsymbol{M}$ 和 $\boldsymbol{S}$ 分别记录了由杂交、变异和选择操作所引起的中间转移概率.

为方便应用，此处引用如下定义和定理。

定义 13.1　$N\times N$ 阶矩阵 $\boldsymbol{P}$ 称为：

① 非负的，若对于所有 $i,j\in\{1,2,\cdots,N\}$ 有 $p_{ij}\geqslant 0$；

② 正的，若对于所有 $i,j\in\{1,2,\cdots,N\}$ 有 $p_{ij}>0$.

非负矩阵 $\boldsymbol{P}$ 称为：

③ 本原的，若存在正整数 k 使 $\boldsymbol{P}^k$ 为正的.

④ 可约的，若通过对行和列采用同样的转换，P 可变换为如下形式

$$\begin{bmatrix}\boldsymbol{B} & \boldsymbol{\phi}\\ \boldsymbol{R} & \boldsymbol{T}\end{bmatrix} \tag{13.2.4}$$

其中 $\boldsymbol{\phi}$ 为零矩阵，$\boldsymbol{B}$ 为 $n\times n$ 阶方阵（$n<N$）.

⑤ 随机的，若对于所有 $i\in\{1,2,\cdots,N\}$ 有 $\sum_{j=1}^{N}p_{ij}=1$.

随机矩阵 $\boldsymbol{P}$ 称为

⑤ 稳定的，若 $\boldsymbol{P}$ 有相同的行.

定理 13.1　设 P 为本原随机矩阵，则

$$\boldsymbol{P}^{\infty}=\lim_{k\to\infty}\boldsymbol{P}^k=1^{\mathrm{T}}\boldsymbol{p}^{\infty}$$

是正的稳定的随机矩阵，其中

$$1^{\mathrm{T}}=(1,1,\cdots,1)^{\mathrm{T}}$$
$$\boldsymbol{p}^{\infty}=\boldsymbol{p}^0\boldsymbol{P}^{\infty}$$

P^0 是初始状态分布，p^∞ 是唯一的，与初始分布无关，且满足 $p_i^\infty > 0$（$i = 1, 2, \cdots, N$）.

定理 13.2　设 P 为可约随机矩阵，可变换为(13.2.4)的形式，其中 B 为 $n \times n$ 阶本原随机矩阵，且 R, T 不为零矩阵，则

$$P^\infty = \lim_{k\to\infty} P^k = \mathbf{1}^T p^\infty$$

是稳定的随机矩阵，其中

$$\mathbf{1}^T = (1, 1, \cdots, 1)^T$$

$$p^\infty = p^0 P^\infty$$

p^0 是初始状态分布，p^∞ 是惟一的，与初始分布无关，且满足

$$p_i^\infty > 0, \qquad \text{当 } 1 \leqslant i \leqslant n$$

$$p_i^\infty = 0, \qquad \text{当 } n < i \leqslant N$$

二、收敛性分析

科学研究、工程实际与国民经济发展中的众多问题可归结为“极大化效益、极小化代价”的模型. 求解这类模型导致寻求某个目标函数在特定区域上的最优解，可描述为(13.2.1)式. 在有关这类模型的相当多的问题中，最优解往往是可以预先已知的. 本节讨论在自然选择下，利用标准遗传算法解决这类最优解为预先已知的优化问题的收敛性问题. 以下定理给出了收敛性分析的结果. 为便于表述，假定问题只有一个全局最优解.

定理 13.3　设种群规模为 m，种群的状态记为$(b_1, b_2, \cdots, b_m)$，其中 $b_i (i = 1, 2, \cdots, m)$是长度为 L 的二进制串. 设适应度函数 $F(b)$的最大值为

$$\max_{b\in E} F(b) = F(b^*) \tag{13.2.5}$$

则当种群的状态达到

$$(b^*, b^*, \cdots, b^*) \tag{13.2.6}$$

时若仍进行变异操作，则标准遗传算法必不收敛. 而若种群达到状态(13.2.6)后，遗传操作不再包含变异运算，则标准遗传算法收敛到$(b^*, b^*, \cdots, b^*)$.

证明　记种群状态为$(b_1, b_2, \cdots, b_m)$. 所有的种群状态构成一个 $1 \times N$ 的行向量，其中 $N = 2^{Lm}$，这个行向量的每个元素由排在一起的 m 个二进制串构成，可表示为

$$\{(b_1^{(1)}, b_2^{(1)}, \cdots, b_m^{(1)}), (b_1^{(2)}, b_2^{(2)}, \cdots, b_m^{(2)}), \cdots, (b_1^{(N)}, b_2^{(N)}, \cdots, b_m^{(N)})\} \tag{13.2.7}$$

注意到通过变异操作可以达到(13.2.7)中的任一状态. 现设经过若干代遗传操作，已达到最优种群状态$(b^*, b^*, \cdots, b^*)$，不妨设其排在状态行向量的第一个分量处，即

$$\{(b_1^{(*)}, b_2^{(*)}, \cdots, b_m^{(*)}), (b_1^{(1)}, b_2^{(1)}, \cdots, b_m^{(1)}), \cdots, (b_1^{(N-1)}, b_2^{(N-1)}, \cdots, b_m^{(N-1)})\} \tag{13.2.8}$$

注意到经过杂交操作，$(b^*, b^*, \cdots, b^*)$仍为其自身，因此相应于状态(13.2.8)的下一代的杂交矩阵可以写成

$$C = \begin{bmatrix} 1 & \phi \\ C_{21} & C_{22} \end{bmatrix} \tag{13.2.9}$$

其中 C 为随机矩阵，且每一行至少有一个正的元素，ϕ 为 $N-1$ 维行向量.

同理，相应于状态(13.2.8)的下一代的选择矩阵可以写成

$$S = \begin{bmatrix} 1 & \phi \\ S_{21} & S_{22} \end{bmatrix} \tag{13.2.10}$$

它也是随机矩阵，且每一列至少有一个正的元素.

如果达到最优种群后仍继续进行变异操作，则种群状态可达到(13.2.7)中的任一状态，于是相应于状态(13.2.8)的下一代的变异矩阵可以写成

$$M=\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \tag{13.2.11}$$

其中 M 为随机矩阵，M_{12} 为 $N-1$ 维行向量，且 $M_{11}, M_{12}, M_{21}, M_{22}$ 的所有元素均大于零.

如果达到最优种群后不再进行变异操作，则相应于状态(13.2.8)的下一代的变异矩阵可以写成

$$M=\begin{bmatrix} 1 & \phi \\ M_{21} & M_{22} \end{bmatrix} \tag{13.2.12}$$

其中 M 为随机矩阵，ϕ 为 $N-1$ 维行向量，且 M_{21}, M_{22} 的所有元素均大于零.

若按(13.2.9)～(13.2.11)式进行演变，则标准遗传算法在由状态(13.2.8)开始进行下一代遗传操作的转移矩阵为

$$\begin{aligned} P=CMS= & \begin{bmatrix} 1 & \phi \\ C_{21} & C_{22} \end{bmatrix}\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}\begin{bmatrix} 1 & \phi \\ S_{21} & S_{22} \end{bmatrix}= \\ & \begin{bmatrix} M_{11} & M_{12} \\ C_{21}M_{11}+C_{22}M_{21} & C_{21}M_{12}+C_{22}M_{22} \end{bmatrix}\begin{bmatrix} 1 & \phi \\ S_{21} & S_{22} \end{bmatrix}= \\ & \begin{bmatrix} M_{11}+M_{12}S_{21} & M_{12}S_{22} \\ C_{21}M_{11}+C_{22}M_{21}+C_{21}M_{12}S_{21}+C_{22}M_{22}S_{21} & C_{21}M_{12}S_{22}+C_{22}M_{22}S_{22} \end{bmatrix}= \\ & \begin{bmatrix} B & D \\ R & T \end{bmatrix} \end{aligned} \tag{13.2.13}$$

显然 P 的各子矩阵 B, D, R, T 都是正的，从而 P 是本原随机矩阵，于是由定理 13.1 有

$$p_i^{\infty}>0 \quad (i=1,2,\cdots,N) \tag{13.2.14}$$

即标准遗传算法不收敛.

若按(13.2.9)，(13.2.10)和(13.2.12)式进行演变，即达到最优种群后就不再进行变异操作，则标准遗传算法进行下一代遗传操作的转移概率矩阵为

$$\begin{aligned} P=CMS= & \begin{bmatrix} 1 & \phi \\ C_{21} & C_{22} \end{bmatrix}\begin{bmatrix} 1 & \phi \\ M_{21} & M_{22} \end{bmatrix}\begin{bmatrix} 1 & \phi \\ S_{21} & S_{22} \end{bmatrix}= \\ & \begin{bmatrix} 1 & \phi \\ C_{21}+C_{22}M_{21} & C_{22}M_{22} \end{bmatrix}\begin{bmatrix} 1 & \phi \\ S_{21} & S_{22} \end{bmatrix}= \\ & \begin{bmatrix} 1 & \phi \\ C_{21}+C_{22}M_{21}+C_{22}M_{22}S_{21} & C_{22}M_{22}S_{22} \end{bmatrix}= \\ & \begin{bmatrix} B & \phi \\ R & T \end{bmatrix} \end{aligned} \tag{13.2.15}$$

显然 P 为可约随机矩阵，且 R, T 不是零矩阵. 若上述规则(13.2.9)，(13.2.10)和(13.2.12)不随时间而变，则由定理 13.2 有

$$p^{\infty}=(p_1^{\infty}, 0, \cdots, 0) \tag{13.2.16}$$

其中 $p_1^{\infty}>0$. 因为 p^{∞} 是一个状态分布，所以应有 $p_1^{\infty}=1$. 于是

$$p^{\infty} = (1,0,\cdots,0) \tag{13.2.17}$$

即标准遗传算法收敛到$(b^*,b^*,\cdots,b^*)$.

上述结果表明,变异操作是影响标准遗传算法的主要因素之一. 在自然选择下,若在遗传操作中始终存在变异操作,则标准遗传算法必不收敛. 而当预先已知全局优化问题最大值时,只要限制达到最优种群后不再进行变异操作,则标准遗传算法必收敛到全局最优解. 这一结论为在自然选择下利用标准遗传算法解决某类最优解为预先已知的优化问题提供了一种理论依据.

关于遗传算法收敛性的研究,还有些其他的结果,但尚无完整的收敛性理论. 目前的这些结果取决于对遗传算法的各种改进以及所采用的数学手段. 现在通常是将群体视为一个随机变数,利用随机过程的模型,通过对随机变数的概率密度分布或均值性质来考察随机变数的演变,以此来进行收敛性分析.

需要指出,尽管在某些条件下,可以证明改进的遗传算法能够收敛至全局最优解,但收敛所需时间可能很长. 这是关于各种改进的遗传算法理论研究中的一个难点,目前尚无任何结果,也是当前最迫切需要解决的问题之一,因为它能从理论上对遗传算法的任何修正形式提供评判标准,以指明改进遗传算法效能的正确方向.

§13.3　基于扩展串的等价遗传算法的收敛性

下面独立地提出一种与标准遗传算法的优化问题等价的遗传算法,旨在从理论上探讨遗传算法收敛的条件与收敛速度等问题.

一、二值串的扩展与优化问题的等价

设染色体的长度为 L,不同染色体的总数为 2^L,各染色体是一个二值串(每个串取其二进制数的值标志). 染色体解码后为 x',其适应度为 $f'(x')$. 不失一般性,可设 $f'(x')>0$(对所有的取值 x'). 今根据这些长度为 L 的染色体定义长度为 $L+1$ 的染色体,即把原来各串前面加 0 或 1. 这样的染色体其总数为 2^{L+1} 个. 对扩展后的二值串的解码仅对各串原来的 L 个二进制数进行,其解码后的结果记为 x,即

$$x=\binom{0}{x'} \quad 或 \quad x=\binom{1}{x'}.$$

定义经扩展的二值串解码后的适应度函数为

$$f(x)=\begin{cases} f(x'), & 当扩展后二值串首位为 0 时 \\ 0, & 当扩展后二值首位为 1 时 \end{cases} \tag{13.3.1}$$

显然

$$\max_x f(x) = \max_{x'} f'(x') \tag{13.3.2}$$

由此可见,这个加长了的染色体解码后适应度的极大值与原问题的极大值是一致的. 所以对原问题寻优相当于对加长染色体后的问题的寻优.

二、染色体的配对与分组

现设问题具有惟一的最大适应度,即仅有一个 x^* 使

$$f(x^*) = \max_x f(x) \tag{13.3.3}$$

不妨设 x^* 对应的染色体的基因的排列为 0,1 相间的，即其基因的排列为 0　1　0　1　0　1…

如上段所述，扩展后的长度为 $L+1$ 的染色体其总数为 2^{L+1} 个. 现在将这 2^{L+1} 个不同的染色体按照 Hamming 距离为 $L+1$ 的原则进行配对，即在每个染色体对中各染色体对应的基因都是不同的. 如第 1 个染色体的第 i 个基因为 1(或为 0)，则第 2 个染色体的第 i 个基因为 0(或为 1). 这个染色体对的适应度的最大值就是第 1 个基因为 0 的染色体的适应度. 按照这样的配对方式，这 2^{L+1} 个不同的染色体可以配成 2^L 个对. 下面对这 2^L 个染色体对进行分组.

第 0 组：$L+1$ 个基因均为 0 和均为 1 的染色体对称为第 0 组；

第 j 组：凡第 $j-1$ 组的各染色体对，各对进行杂交后，得到的所有的新的染色体对的集合称为第 j 组.

一般地，可分为 $L+1$ 个组(从第 0 组到第 L 组)，第 j 组的染色体对的数目为 C_L^j 个.

三、不考虑适应度的杂交

我们把上述分组后得到的每一组看成一种状态，于是共有 $L+1$ 个状态，即在同一组内的各不同染色体对被视为同一种状态. 将群体内的染色体对看成一个随机变数 ζ(以 $0,1,\cdots,L$ 标志其 $L+1$ 个状态)，它具有概率分布. 考虑单点杂交操作，杂交后群体内状态的概率分布将发生演变，这种演变形成了随机过程 $\zeta(t), t=1,2,\cdots$. 为了掌握群体内状态的概率分布，我们必须首先了解由杂交引起的转移概率密度 p_{ij}，此处

$$p_{ij}=p\{\zeta(t+1)=i\mid\zeta(t)=j\}\qquad(i,j=0,1,\cdots,L)\tag{13.3.4}$$

现在来确定 p_{ij}. 从分组的方法可以看出，第 j 组中的染色体对，必由第 $j-1$ 组的染色体杂交而得，而第 j 组中染色体对杂交后的染色体对只可能在第 $j-1$ 组或第 $j+1$ 组中.

第 j 组杂交后得出的染色体对的总数为 LC_L^j. 将第 j 组转向第 $j-1$ 组的染色体对的总数记为 $S_{j-1,j}(j=1,2,\cdots,L)$. 定义

$$S_{-1,0}=0\tag{13.3.5}$$

将第 j 组转向第 $j+1$ 组的染色体对的总数记为 $S_{j+1,j}$，于是有：

$$S_{j+1,j}=LC_L^j-S_{j-1,j}\qquad(j=0,1,\cdots,L-1)\tag{13.3.6}$$

因此，第 j 组转向第 $j-1$ 组的概率为

$$p_{j-1,j}=\frac{S_{j-1,j}}{LC_L^j}\quad(j=1,2,\cdots,L)\tag{13.3.7}$$

第 j 组转向第 $j-1$ 组的概率为

$$p_{j+1,j}=\frac{S_{j+1,j}}{LC_L^j}=\frac{LC_L^j-S_{j-1,j}}{LC_L^j}=1-p_{j-1,j}\qquad(j=0,1,\cdots,L-1)\tag{13.3.8}$$

注意到：

$$S_{j,j+1}=S_{j+1,j}\qquad(j=0,1,\cdots,L)\tag{13.3.9}$$

于是由(13.3.5)～(13.3.9)式可以递推求得第 j 组转向第 $j-1$ 组和第 $j+1$ 组的概率. 进一步定义

$$p_{ij}=\begin{cases}p_{j-1,j}, & 当\ i=j-1 \quad (j=1,2,\cdots,L)\\ p_{j+1,j}, & 当\ i=j+1 \quad (j=0,1,\cdots,L-1)\\ 0, & 当\ i\neq j-1, i\neq j+1 \quad (j=0,1,\cdots,L)\end{cases}\tag{13.3.10}$$

可得转移概率矩阵 $p=(p_{ij})$. 例如，$L=2$ 和 $L=3$ 时的转移概率矩阵相应为

$$p=\begin{pmatrix}0 & \frac{1}{2} & 0\\ 1 & 0 & 1\\ 0 & \frac{1}{2} & 0\end{pmatrix},p=\begin{pmatrix}0 & \frac{1}{3} & 0 & 0\\ 1 & 0 & \frac{2}{3} & 0\\ 0 & \frac{2}{3} & 0 & 1\\ 0 & 0 & \frac{1}{3} & 0\end{pmatrix}$$

由以上讨论可见，p 是一个三对角矩阵，且显然有

$$p_{ij} \geqslant 0 \qquad (i,j=0,1,\cdots,L) \tag{13.3.11}$$

$$\sum_{i=0}^{L} P_{ij} = p_{j-1,j} + p_{j+1,j} = 1 \tag{13.3.12}$$

因此，随机变数 ζ 的状态概率密度向量 $\{\zeta^0\}=\{\zeta\}$ 经不断变换

$$\{\zeta^{t+1}\}=p\{\zeta^{(t)}\} \tag{13.3.13}$$

后转移为 $\{\zeta^{(1)}\},\{\zeta^{(2)}\},\cdots,\{\zeta^{(n)}\},\cdots$，它是一个有限齐次马尔可夫链.

四、考虑适应度的杂交

现在研究杂交以后考虑适应度的情形. 研究第 j 组，经杂交后转移到第 $j-1$ 组的概率为 $p_{j-1,j}$；经杂交后转移到第 $j+1$ 组的概率为 $p_{j+1,j}$. 以 $x_{j,i}$ 记第 j 组第 i 个染色体对解码后的值 $(i=1,2,\cdots,C_L^j)$. 每个染色体对（不妨设第 i 个染色体对）经杂交后必位于第 $j-1$ 组或第 $j+1$ 组，它等概率地被杂交成 $M(j),N(j)$ 个第 $j-1$ 组和第 $j+1$ 组的染色体对. 它们解码后的值记为：

$$x_{j,i,j-1,k} \quad (k=1,2,\cdots,M(j);i=1,2,\cdots,C_L^j)$$
$$x_{j,i,j+1,k} \quad (k=1,2,\cdots,N(j);i=1,2,\cdots,C_L^j)$$

其中：

$$M(j)+N(j)=L$$

记 $f(x_{j,i})\geqslant f(x_{j,i,j-1,k})$ 的个数为 $l_1(j,i)$，于是 $f(x_{j,i})<f(x_{j,i,j-1,k})$ 的个数为 $M(j)-l_1(j,i)$. 从而第 j 组第 i 个染色体对向第 $j-1$ 组转移的概率为 $1-\frac{l_1(j,i)}{M(j)}$，不向第 $j-1$ 组转移的概率为 $\frac{l_1(j,i)}{M(j)}$. 注意到第 j 组的染色体对共有 C_L^j 个，于是有第 j 组向第 $j-1$ 组转移（考虑适应度后）的概率为

$$p_{j-1,j}\Big(1-\frac{1}{C_L^j}\sum_{i=1}^{C_L^j}\frac{l_1(j,i)}{M(j)}\Big) \equiv p_{j-1,j}q_{j-1,j} \tag{13.3.14}$$

第 j 组不向第 $j-1$ 组转移（考虑适应度后）的概率为：

$$p_{j-1,j}\frac{1}{C_L^j}\sum_{i=1}^{C_L^j}\frac{l_1(j,i)}{M(j)} = p_{j-1,j}(1-q_{j-1,j}) \tag{13.3.15}$$

同样，设 $f(x_{j,i})\leqslant f(x_{j,i,j+1,k})$ 的个数为 $l_2(j,i)$，于是 $f(x_{j,i})>f(x_{j,i,j+1,k})$ 的个数为 $N(j)-l_2(j,i)$. 从而第 j 组第 i 个染色体对向第 $j+1$ 组转移的概率为 $\frac{l_2(j,i)}{N(j)}$，不向第 $j+1$ 组转移的概率为 $1-\frac{l_2(j,i)}{N(j)}$. 最终有第 j 组向第 $j+1$ 组转移（考虑适应度后）的概率为：

$$p_{j+1,j}\frac{1}{C_L^j}\sum_{i=1}^{C_L^j}\frac{l_2(j,i)}{N(j)}=p_{j+1,j}q_{j+1,j} \tag{13.3.16}$$

第 j 组不向第 $j+1$ 组转移(考虑适应度后)的概率为:

$$p_{j+1,j}\left(1-\frac{1}{C_L^j}\sum_{i=1}^{C_L^j}\frac{l_2(j,i)}{N(j)}\right)=p_{j+1,j}(1-q_{j+1,j}) \tag{13.3.17}$$

于是可得考虑适应度后的转移概率矩阵为 $p_f=p_{ij}^f$,其中:

$$p_{ij}^f=\begin{cases}p_{j-1,j}q_{j-1,j}, & 当\ i=j-1 \quad (j=1,2,\cdots,L)\\ p_{j-1,j}(1-q_{j-1,j})+p_{j+1,j}(1-q_{j+1,j}), & 当\ i=j \quad (j=1,2,\cdots,L)\\ p_{j+1,j}q_{j+1,j}, & 当\ i=j+1 \quad (j=0,1,\cdots,L-1)\\ 0, & 当\ i\ 取其他值 \quad (j=0,1,\cdots,L)\end{cases} \tag{13.3.18}$$

由此可见,p_f 也是一个三对角矩阵,且显然有

$$p_{ij}^f\geqslant 0 \qquad (i,j=0,1,\cdots,L) \tag{13.3.19}$$

$$\sum_{i=0}^{L}p_{ij}^f=1 \tag{13.3.20}$$

因此,随机变数 ζ 的状态概率密度向量 $\{\zeta_f^{(0)}\}=\{\zeta\}$ 经不断变换

$$\{\zeta_f^{(t+1)}\}=p_t\{\zeta_t^{(t)}\} \tag{13.3.21}$$

后转移为 $\{\zeta_t^{(1)}\},\{\zeta_t^{(2)}\},\cdots,\{\zeta_f^{(n)}\},\cdots$,它也是一个有限齐次马尔可夫链.

由于已设定第 $L+1$ 组是最优解,故有

$$p_{iL}=p_{iL}^f=\begin{cases}1, & 当\ i=L\\ 0, & 当\ i\neq L\end{cases} \tag{13.3.22}$$

这样,演变过程(不考虑适应度或考虑适应度)可以看成有限齐次马尔可夫链.

五、收敛性分析

由上述讨论可见,本文提出的等价遗传算法可以描述为 $L+1$ 个状态的齐次马尔可夫链,其最优点位于 $\zeta=L$ 处.具体写成

$$\{\zeta^{(t+1)}\}=p\{\zeta^{(t)}\} \quad (不考虑适应度) \tag{13.3.23}$$

$$\{\zeta_f^{(t+1)}\}=p\{\zeta_f^{(t)}\} \quad (考虑适应度) \tag{13.3.24}$$

p,p_f 都是三对角矩阵,具有形式

$$\begin{bmatrix}c_0 & a_0 & & & & & 0\\ b_1 & c_1 & a_1 & & & & \\ & b_2 & c_2 & a_2 & & & \\ & & & \ddots & & & \\ & & & & b_{L-1} & c_{L-1} & a_{L-1}\\ 0 & & & & & b_L & c_L\end{bmatrix}$$

其中次对角元 $a_i(i=0,1,\cdots,L-1)$,$b_i(i=1,2,\cdots,L)$ 与对角元 $c_i(i=0,1,\cdots,L)$ 满足:

$$\begin{cases}a_i\geqslant 0,b_i\geqslant 0,\ c_i\geqslant 0\\ c_0+b_1=1\\ a_i+c_{i+1}+b_{i+2}=1\\ a_{L-1}+c_L=1\end{cases} \tag{13.3.25}$$

p 与 p_f 之不同是，在 p 中 $c_i=0(i=1,2,\cdots,L)$. 对于已经设定第 $L+1$ 组是极大的情况则将有：

$$c_L=1, \quad a_{L+1}=0 \tag{13.3.26}$$

现在设：

$$\begin{cases} a_i>0 & (i=0,1,\cdots,L-2) \quad a_{L-1}=0 \\ b_i>0 & (i=1,2,\cdots,L) \\ c_i\geqslant 0 & (i=0,1,2,\cdots,L-1) \quad c_L=1 \\ c_0+b_1=1 & \\ a_i+c_{i+1}+b_{i+2}=1 & (i=0,1,\cdots,L-2) \end{cases} \tag{13.3.27}$$

于是可以证明，无论 p 或 p_f 均有 $L+1$ 个实的单特征值 $\lambda_0,\lambda_1,\cdots,\lambda_L$，且 $|\lambda_i|<1(i=0,1,\cdots,L-1)$，$\lambda_L=1$，(证明略).

于是存在非奇异矩阵 $\boldsymbol{\Phi}$ 与 $\boldsymbol{\Psi}$ 使

$$\boldsymbol{p}=\boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^{-1} \tag{13.3.28}$$

$$\boldsymbol{p}_f=\boldsymbol{\Psi}\boldsymbol{\Lambda}_f\boldsymbol{\Psi}^{-1} \tag{13.3.29}$$

其中 $\boldsymbol{\Lambda}$ 与 $\boldsymbol{\Lambda}_f$ 分别是由 $\boldsymbol{p}$ 与 $\boldsymbol{p}_f$ 的全部特征值构成的对角矩阵.

设群体给定了初始分布 $\{\zeta^{(0)}\}$，则有

$$\begin{aligned} \{\zeta^{(t)}\} &= \boldsymbol{P}\{\zeta^{(t-1)}\} = \\ &\boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^{-1}\{\zeta^{(t-1)}\} = \\ &\boldsymbol{\Phi}\boldsymbol{\Lambda}^2\boldsymbol{\Phi}^{-1}\{\zeta^{(t-2)}\} = \\ &\cdots\cdots = \\ &\boldsymbol{\Phi}\boldsymbol{\Lambda}^t\boldsymbol{\Phi}^{-1}\{\zeta^{(0)}\} \end{aligned} \tag{13.3.30}$$

及

$$\{\zeta_f^{(t)}\}=\boldsymbol{\Psi}\boldsymbol{\Lambda}_f^t\boldsymbol{\Psi}^{-1}\{\zeta^{(0)}\} \tag{13.3.31}$$

于是

$$\lim_{t\to\infty}\{\zeta_f^{(t)}\}=\boldsymbol{\Psi}\lim_{t\to\infty}\boldsymbol{\Lambda}_f^t\boldsymbol{\Psi}^{-1}\{\zeta^{(0)}\}= \boldsymbol{\Psi}\begin{pmatrix} 0 & & & & 0 \\ & \ddots & & & \\ & & 0 & & \\ & & & \ddots & \\ 0 & & & & 1 \end{pmatrix}\boldsymbol{\Psi}^{-1}\{\zeta^{(0)}\} \tag{13.3.32}$$

定理 13.4 若转移概率密度矩阵满足(13.3.27)式，则按本节提出的遗传算法可收敛到最优解，即：

$$\lim_{t\to\infty}\{\zeta_f^{(t)}\}=\begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \tag{13.3.33}$$

证明 注意到当 $a_{L-1}=0$ 时，$\lambda=1$ 是一个特征值，其相应的右特征向量可取为 $(0,0,\cdots,0,1)^{\mathrm{T}}$，左特征向量可取为 $(1,1,\cdots,1)$，从而 $\boldsymbol{\Psi}$ 和 $\boldsymbol{\Psi}^{-1}$ 分别具有如下形式：

$$\Psi=\begin{pmatrix}\Psi_{00} & \cdots & \Psi_{0,L-1} & 0\\ \vdots & & \vdots & \vdots\\ \Psi_{L-1,0} & \cdots & \Psi_{L-1,L-1} & 0\\ \Psi_{L,0} & \cdots & \Psi_{L,L-1} & 1\end{pmatrix},\Psi^{-1}=\begin{pmatrix}\Psi_{00}^{-1} & \cdots & \Psi_{0,L-1}^{-1} & 0\\ \vdots & & \vdots & \vdots\\ \Psi_{L-1,0}^{-1} & \cdots & \Psi_{L-1,L-1}^{-1} & 0\\ 1 & \cdots & 1 & 1\end{pmatrix}$$

于是由(13.3.32)式可得 $\zeta_{ji}^{(t)}\to 0\quad(i=0,1,\cdots,L-1)$，$\zeta_{jL}^{(t)}\to 1$. 证毕.

下面讨论平均到达最优点的次数.记随机变数 $\zeta_{jL}^{(t)}$ 的状态概率密度为：

$$\{\zeta_j^{(t)}\}=\{\zeta_{j0}^{(t)},\zeta_{j1}^{(t)},\zeta_{jL}^{(t)}\}^{\mathrm T}$$

记 Ψ 的矩阵元为 Ψ_{ij}，并设 $\Psi_{LL}=1$. 记 Ψ^{-1} 的矩阵元为 Ψ_{ij}^{-1}，于是 $\Psi_{L0}^{-1}=\Psi_{L1}^{-1}=\cdots=\Psi_{LL}^{-1}=1$. 为方便计，仍将 A_f 的对角元素记为 $\lambda_i(i=0,1,\cdots,L)$，从而(13.3.31)式可写成：

$$\begin{pmatrix}\zeta_{f0}^{(t)}\\ \zeta_{f1}^{(t)}\\ \vdots\\ \zeta_{fL}^{(t)}\end{pmatrix}=\begin{pmatrix}\Psi_{00} & \Psi_{01} & \cdots & \Psi_{0L}\\ \Psi_{10} & \Psi_{11} & \cdots & \Psi_{1L}\\ \vdots & \vdots & & \vdots\\ \Psi_{L0} & \Psi_{L1} & \cdots & \Psi_{LL}\end{pmatrix}\begin{pmatrix}\lambda_0^t & & 0\\ & \lambda_1^t & \\ & & \ddots\\ 0 & & \lambda_L^t\end{pmatrix}\cdot$$

$$\begin{pmatrix}\Psi_{00}^{-1} & \Psi_{01}^{-1} & \cdots & \Psi_{0L}^{-1}\\ \Psi_{10}^{-1} & \Psi_{11}^{-1} & \cdots & \Psi_{1L}^{-1}\\ \vdots & \vdots & & \vdots\\ \Psi_{L0}^{-1} & \Psi_{L1}^{-1} & \cdots & \Psi_{LL}^{-1}\end{pmatrix}\begin{pmatrix}\zeta_0^{(0)}\\ \zeta_1^{(0)}\\ \vdots\\ \zeta_L^{(0)}\end{pmatrix}=$$

$$\begin{pmatrix}\sum_{j=0}^{L}\lambda_j^t\Psi_{0j}\sum_{j=0}^{L}\Psi_{ji}^{-1}\zeta_i^{(0)}\\ \sum_{j=0}^{L}\lambda_j^t\Psi_{1j}\sum_{j=0}^{L}\Psi_{ji}^{-1}\zeta_i^{(0)}\\ \vdots\\ \sum_{j=0}^{L}\lambda_j^t\Psi_{Lj}\sum_{j=0}^{L}\Psi_{ji}^{-1}\zeta_i^{(0)}\end{pmatrix}\tag{13.3.34}$$

于是有

$$\zeta_{fL}^{(t)}=1+\sum_{j=0}^{L-1}\lambda_i^t\Psi_{Lj}\sum_{i=0}^{L}\Psi_{ji}^{-1}\zeta_i^{(0)}\tag{13.3.35}$$

从面恰为一次到达最优点的概率为

$$1+\sum_{j=0}^{L-1}\lambda_j\Psi_{Lj}\sum_{i=0}^{L}\Psi_{ji}^{-1}\zeta_i^{(0)}\tag{13.3.36}$$

恰为 k 次到达最优点的概率为

$$\sum_{j=0}^{L-1}(\lambda_j^k-\lambda_j^{k-1})\Psi_{Lj}\sum_{i=0}^{L}\Psi_{ji}^{-1}\zeta_i^{(0)}\quad(k>1)\tag{13.3.37}$$

于是平均到达最优点的次数为

$$T=1+\sum_{j=0}^{L-1}\lambda_j\Psi_{Lj}\sum_{i=0}^{L}\Psi_{ji}^{-1}\zeta_i^{(0)}+2\sum_{j=0}^{L-1}(\lambda_j^2-\lambda_j)\sum_{i=0}^{L}\Psi_{ji}^{-1}\zeta_i^{(0)}+\cdots+$$
$$k\sum_{j=0}^{L-1}(\lambda_j^k-\lambda_j^{k-1})\Psi_{Lj}\sum_{i=0}^{L}\Psi_{ji}^{-1}\zeta_i^{(0)}+\cdots=$$

$$1-\sum_{k=1}^{\infty}\sum_{j=0}^{L-1}\lambda_j^k\Psi_{Lj}\sum_{i=0}^{L}\Psi_{ji}^{-1}\zeta_i^{(0)}=$$

$$1-\sum_{j=0}^{L-1}\frac{\lambda_j}{1-\lambda_j}\Psi^{Lj}\sum_{i=0}^{L}\Psi_{ji}^{-1}\zeta_i^{(0)} \tag{13.3.38}$$

特别地，若取 $\zeta_0^{(0)}=1,\zeta_i^{(0)}=0(i=1,2,\cdots,L)$，则平均到达最优点的次数为：

$$T=1-\sum_{j=0}^{L-1}\frac{\lambda_j}{1-\lambda_j}\Psi_{Lj}\Psi_{j0}^{-1} \tag{13.3.39}$$

当平均到达次数 $T<2^L$ 时遗传算法才比穷举法更为有效.

六、结论与讨论

(1) 本文提出的等价遗传算法的运行是按组进行的，不同于常规遗传算法. 用该算法可以收敛到最优解(考虑到适应度)，且平均到达次数是有限的.

(2) 若次对角线上某个 $b_i=0$，则第 $i-1$ 组无法转移到第 i 组. 因此若群体的初始状态分布只有前 $i-1$ 组的值不为零，则绝无可能收敛到最优解.

这种算法中没有考虑变异. 但有了变异却可以使群体中的概率分布随机地改变，使群体的分布达到按序号排列较大的组中，从而使收敛的可能性加大.

若 $b_i\neq0$ 则遗传算法的收敛性与初始分布是无关的，但收敛速度与初始分布是有关系的. 总体说来，当存在接近于 1 的特征值时收敛就比较慢.

上述算法的数学模型是标准的有限齐次马尔可夫过程，对其收敛性及收敛速度进行了讨论，然而与实际操作时的标准过程还有较大差距.

§13.4　选择和变异操作下遗传算法的收敛性

下面来分析选择和变异操作下遗传算法的收敛性问题. 首先以较简洁的方式给出连续空间中群体概率密度演变公式的证明，并提出离散空间中群体的演变过程收敛到全局最优状态的一个充分条件，从而得到对遗传算法的实现具有一定指导意义的结果. 这在一定程度上为实现自适应调节变异操作，以保证遗传算法收敛到全局最优解提供了理论依据.

一、连续空间中群体概率密度演变公式

考虑优化问题

$$arg\ \max_x g(\boldsymbol{x}) \tag{13.4.1}$$

其中 $\boldsymbol{x}\in \boldsymbol{F}$ 为实参数向量. $\boldsymbol{F}$ 为可行域，$g(\boldsymbol{x})$ 为适应度函数，假定其满足如下条件：

(1) $g(\boldsymbol{x})$ 只有有限个全局最大值；

(2) $0<g_{\min}\leqslant g(\boldsymbol{x})\leqslant g_{\max}<\infty,\ \forall\ \boldsymbol{x}\in \boldsymbol{F}$；

(3) $g(\boldsymbol{x})$ 有有限个不连续点.

以下仅讨论在选择和变异操作下的遗传算法. 设在时刻 k，群体具有 N 个随机向量，记为 $\boldsymbol{X}_k=\{x_k^1,x_k^2,\cdots,x_k^N,\}$，其概率密度记为 $f_{x_k}(x_k^1,x_k^2,\cdots,x_k^N)$. 群体中任意一个向量 x_k^j 的概率密度记为 $f_{x_k^j}(\boldsymbol{x})(j=1,2,\cdots,N)$，设各 x_k^j 是同分布的. x_k^j 经选择后记为 $\boldsymbol{x}'^i_k$，其概率密度记为 $f_{x'^i_k}(\boldsymbol{x})(i,j=1,2,\cdots,N)$. x'^i_k 经变异后记为 $\boldsymbol{x}^i_{k+1}$，其概率密度记为 $f^i_{x_{k+1}}(\boldsymbol{x})(i=1,2,\cdots,N)$. 为便于讨论，以下将群体 $\boldsymbol{X}_k$ 的取值以 $y^1,y^2,\cdots,y^N$ 记之，$\boldsymbol{x}'^i_k$ 的取值以 z 记之，$\boldsymbol{x}^i_{k+1}$ 的取值以 $\boldsymbol{x}$ 记之. 设选择运算按下述公式进行，有

$$P_r\{x'^i_k = x^j_k \mid X_k\} = \frac{g(x^j_k)}{\sum_{i=1}^{N} g(x^i_k)} \quad (i,j = 1,2,\cdots,N) \tag{13.4.2}$$

设变异运算以相同的条件概率密度函数

$$f_{x^i_{k+1}|x'^i_k}(x|z) = f_{\omega_k}(x|z) \quad (i=1,2,\cdots,N) \tag{13.4.3}$$

独立地作用于群体中每个元素，且条件概率密度函数满足：

$$\sup_{x,z\in F} f_{\omega_k}(x \mid z) \leqslant M_0 < \infty \tag{13.4.4}$$

对于没有杂交操作的遗传算法，其连续空间中群体的概率密度公式有如下定理.

定理 13.5　若算法按照(13.4.2)式和(13.4.3)式运行，则当 $N\to\infty$ 时，无杂交操作的遗传算法的随机向量序列 $\{x_k\}_{k=0}^{\infty}(x_k \in F)$ 的概率密度在以概率 1 收敛的意义下，具有如下形式

$$f_{x_{k+1}}(x) = \frac{\int_F f_{x_k}(y) g(y) f_{\omega_k}(x \mid y) \mathrm{d}y}{\int_F f_{x_k}(y) g(y) \mathrm{d}y} \tag{13.4.5}$$

下面我们给出关于定理 13.5 的简洁证明.

证明　因为选择运算是按(13.4.2)式进行的，所以选择后群体的第 i 个元素的概率密度在时刻 k 为

$$f_{x'^i_k|X_k}(z \mid y^1, y^2, \cdots, y^N) = \frac{\sum_{j=1}^{N} \delta(z - y^j) g(y^j)}{\sum_{l=1}^{N} g(y^l)}$$

$$\forall i,j = 1,2,\cdots,N, \forall z, y^1, y^2, \cdots, y^N \in F \tag{13.4.6}$$

其中 δ 为 Dirac 函数. 又因为对于群体的每个元素，变异操作独立地按(13.4.3)式进行，故有

$$f_{x^i_{k+1}|X_k}(x \mid y^1, y^2, \cdots, y^N) = \int_F f_{x^i_{k+1}|x'^i_k}(x \mid z) f_{x'^i_k|X_k}(z \mid y^1, y^2, \cdots, y^N) \mathrm{d}z =$$

$$\int_F f_{\omega_k}(x \mid z) \frac{\sum_{j=1}^{N} \delta(z - y^j) g(y^j)}{\sum_{l=1}^{N} g(y^l)} =$$

$$\frac{\sum_{j=1}^{N} g(y^j) f_{\omega_k}(x \mid y^j)}{\sum_{l=1}^{N} g(y^l)}, \forall i = 1,2,\cdots,N \tag{13.4.7}$$

利用全概率定理，有

$$f_{x^i_{k+1}}(x) = \int_{F^N} f_{x^i_{k+1}|X_k}(x \mid y^1, \cdots, y^N) f_{X_k}(y^1, \cdots, y^N) \mathrm{d}y^1 \cdots \mathrm{d}y^N =$$

$$\int_F \frac{\frac{1}{N}\sum_{j=1}^{N} g(y^j) f_{\omega_k}(x \mid y^j)}{\frac{1}{N}\sum_{l=1}^{N} g(y^l)} f_{X_k}(y^1, \cdots, y^N) \mathrm{d}y^1 \cdots \mathrm{d}y^N \triangleq$$

$$f_{x_{k+1}^i}(x,N),\quad \forall i=1,2,\cdots,N \tag{13.4.8}$$

对于固定的 x，$f_{\omega_k}(x|y^j)$是确定的函数. 设 $y^1,\cdots,y^N,\cdots$可以独立选取，于是$\{g(y^j)\}_{j=1}^{\infty}$是相互独立相同分布的随机变量序列，故对于固定的 x，$\{f_{\omega_k}(x|y^j)g(y^j)\}_{j=1}^{\infty}$也是相互独立相同分布的随机变量序列. 由柯尔莫哥洛夫强大数定律，有：

$$\frac{1}{N}\sum_{l=1}^{N}g(y^l)\xrightarrow{\text{a.s.}}\int_F f_{x_k}(y)g(y)\mathrm{d}y\triangleq E_2 \tag{13.4.9}$$

$$\frac{1}{N}\sum_{j=1}^{N}g(y^j)f_{\omega_k}(x\mid y^j)\xrightarrow{\text{a.s.}}\int_F f_{x_k}(y)g(y)f_{\omega_k}(x\mid y)\mathrm{d}y\triangleq E_1(x) \tag{13.4.10}$$

以下往证

$$f_{x_{k+1}^i}(x,N)\xrightarrow{\text{a.s.}}\frac{E_1(x)}{E_2} \tag{13.4.11}$$

首先，注意到

$$\begin{aligned}
&\left|f_{x_{k+1}^i}(x,N)-\frac{E_1(x)}{E_2}\right|=\\
&\left|\int_{F^N}\frac{\frac{1}{N}\sum_{j=1}^{N}g(y^j)f_{\omega_k}(x\mid y^j)}{\frac{1}{N}\sum_{l=1}^{N}g(y^l)}f_{x_k}(y^1,\cdots,y^N)\mathrm{d}y^1\mathrm{d}y^N-\int_{F^N}\frac{E_1(x)}{E_2}f_{x_k}(y^1,\cdots,y^N)\mathrm{d}y^1\cdots\mathrm{d}y^N\right|=\\
&\left|\int_{F^N}\left[\frac{\frac{1}{N}\sum_{j=1}^{N}g(y^j)f_{\omega_k}(x\mid y^j)}{\frac{1}{N}\sum_{l=1}^{N}g(y^l)}-\frac{E_1(x)}{E_2}\right]f_{x_k}(y^1,\cdots,y^N)\mathrm{d}y^1\cdots\mathrm{d}y^N\right|\leqslant\\
&\int_{F^N}\left|\frac{\frac{1}{N}\sum_{j=1}^{N}g(y^j)f_{\omega_k}(x\mid y^j)}{\frac{1}{N}\sum_{l=1}^{N}g(y^l)}-\frac{E_1(x)}{E_2}\right|f_{x_k}(y^1,\cdots,y^N)\mathrm{d}y^1\cdots\mathrm{d}y^N
\end{aligned} \tag{13.4.12}$$

再注意到

$$\begin{aligned}
&\left|\frac{\frac{1}{N}\sum_{j=1}^{N}g(y^j)f_{\omega_k}(x\mid y^j)}{\frac{1}{N}\sum_{l=1}^{N}g(y^l)}-\frac{E_1(x)}{E_2}\right|\leqslant\\
&\frac{1}{g_{\min}E_2}\left|E_2\frac{1}{N}\sum_{j=1}^{N}g(y^j)f_{\omega_k}(x\mid y^j)-E_2E_1(x)+E_2E_1(x)-E_1(x)\frac{1}{N}\sum_{l=1}^{N}g(y^l)\right|\leqslant\\
&\frac{1}{g_{\min}E_2}\left\{|E_2|\cdot\left|\frac{1}{N}\sum_{j=1}^{N}g(y^j)f_{\omega_k}(x\mid y^j)-E_1(x)\right|+|E_1(x)|\cdot\left|E_2-\frac{1}{N}\sum_{l=1}^{N}g(y^l)\right|\right\}\\
&\xrightarrow{\text{a.s.}}0
\end{aligned} \tag{13.4.13}$$

于是有(13.4.11)式成立，即当 $N\to\infty$时(13.4.5)式成立. 证毕.

定理 13.5 给出了在连续空间中具有无穷大种群规模的遗传算法(无杂交操作)的群体的概率密度演变公式. 下面将(13.4.5)式推广应用于离散空间中无杂交操作的遗传算法，并给出群体演变过程收敛到全局最优状态的一个充分条件.

二、离散空间中无杂交操作的遗传算法的收敛性分析

对于离散情形，仍设种群规模为无穷大．将群体中的一种相同的个体认为是一种状态，以记个体的长度，以 $i=1,2,\cdots,M$ 记个体的各种状态．于是，在以概率 1 收敛的意义下，连续空间中在选择与变异作用下种群演变的概率密度表达式可以在离散空间写为：

$$f_i^{(k+1)}=\frac{\sum_{j=1}^{M}m_{ij}^{(k)}f_j^{(k)}g_j}{\sum_{j=1}^{M}f_l^{(k)}g_l}\quad(i=1,2,\cdots,M)\tag{13.4.14}$$

其中：

$$m_{ij}^{(k)}=(1-p_m^{(k)})^{L-d}(p_m^{(k)})^d\quad(i,j=1,2,\cdots,M)\tag{13.4.15}$$

$p_m^{(k)}$ 为随演变过程变化的变异概率，d 为状态 i,j 之间的海明距离．(13.4.14)式用矩阵写出为：

$$\{f^{k+1}\}=[m_{ij}^{(k)}]\mathrm{diag}\left[\frac{g_j}{\{g\}^{\mathrm{T}}\{f^{(k)}\}}\{f^{(k)}\}\right]\tag{13.4.16}$$

当变异概率较小时，由变异引起的转移概率矩阵$[m_{ij}^{(k)}]$可以写成：

$$[m_{ij}^{(k)}]=\begin{bmatrix}1-\varepsilon_{11}^{(k)} & \varepsilon_{12}^{(k)} & \cdots & \varepsilon_{1M}^{(k)}\\ \varepsilon_{21}^{(k)} & 1-\varepsilon_{22}^{(k)} & \cdots & \varepsilon_{2M}^{(k)}\\ \vdots & \vdots & \vdots & \\ \varepsilon_{M1}^{(k)} & \varepsilon_{M2}^{(k)} & \cdots & 1-\varepsilon_{MM}^{(k)}\end{bmatrix}\tag{13.4.17}$$

其中 $\varepsilon_{ij}^{(k)}>0(i,j=1,2,\cdots,M)$ 为小参数，且满足 $\sum_{j\neq i}^{M}\varepsilon_{ij}^{(k)}=\varepsilon_{ij}^{(k)}(i=1,2,\cdots,M)$．可以证明，当变异概率随演变过程变得充分小时，无杂交操作的遗传算法的演变过程收敛到全局最优状态．下述定理给出了收敛的一个充分条件．

定理 13.6　设离散空间中无穷大规模种群的状态的概率密度服从(13.4.14)式，设变异概率随演变过程变得充分小，使由变异引起的转移概率矩阵$[m_{ij}^{(k)}]$$(k=0,1,2,\cdots)$可以写成(13.4.17)式的形式．设 i^* 对应于使适应度达全局最优的状态，即

$$g_{i^*}=\max_{1\leqslant j\leqslant M}g_j\tag{13.4.18}$$

并设 $g_{i^{**}}$ 为次大适应度值，即 $g_i\leqslant g_{i^{**}}\leqslant g_{i^*}$ $(j=1,2,\cdots,M;i\neq i^*)$．于是当

$$p_m^{(k)}<\frac{1-f_{i^*}^{(k)}}{L}\left(1-\frac{g_{i^{**}}}{g_{i^*}}\right.\quad(k=0,1,2,\cdots)\tag{13.3.19}$$

时，有

$$f_{i^*}^{k+1}>f_{i^*}^{(k)}\quad(k=0,1,2,\cdots)\tag{13.4.20}$$

且有

$$\lim_{k\to\infty}f_{i^*}^{(k)}=1\tag{13.4.21}$$

证明　注意到

$$\frac{f_{i^*}^{(k+1)}}{f_{i^*}^{(k)}}=\frac{\sum_{j=1}^{N}m_{i^*j}^{(k)}g_jf_j^{(k)}}{f_{i^*}^{M}\sum_{l=1}^{M}f_l^{(k)}g_l}=\frac{\sum_{j=1}^{M}m_{i^*j}^{(k)}g_jf_j^{(k)}/f_{i^*}^{(k)}}{\sum_{l=1}^{M}f_l^{(k)}g_l}=$$

$$\frac{m_{i^*i^*}^{(k)} g_{i^*} + \sum_{j\neq i^*}^{M} \varepsilon_{i^*j}^{(k)} g_i f_j^{(k)} / f_{i^*}^{(k)}}{\sum_{l=1}^{M} f_l^{(k)} g_l} =$$

$$\left(m_{i^*i^*}^{(k)} + \sum_{j\neq i^*}^{M} \varepsilon_{i^*j}^{(k)} \cdot \frac{g_j}{g_{i^*}} \cdot \frac{f_j^{(k)}}{f_{i^*}^{(k)}}\right) / \left(\sum_{l=1}^{M} f_l^{(k)} \frac{g_l}{g_{i^*}}\right) \geqslant$$

$$\frac{m_{i^*i^*}^{(k)}}{\sum_{l=1}^{M} f_l^{(k)} g_l / g_{i^*}} \tag{13.4.22}$$

再注意到

$$m_{i^*i^*}^{(k)} = 1 - \varepsilon_{i^*i^*}^{(k)} = (1 - p_m^{(k)}) > 1 - Lp_m^{(k)} \tag{13.4.23}$$

于是有

$$\varepsilon_{i^*i^*}^{(k)} < Lp_m^{(k)} \tag{13.4.24}$$

从而当

$$p_m^{(k)} < \frac{1 - f_{i^*}^{(k)}}{L}\left(1 - \frac{g_{i^{**}}}{g_{i^*}}\right) \tag{13.4.25}$$

时，有

$$\varepsilon_{i^*i}^{(k)} < Lp_m^{(k)} < (1 - f_{i^*}^{(k)})\left(1 - \frac{g_{i^{**}}}{g_{i^*}}\right) \tag{13.4.26}$$

故有

$$\frac{f_{i^*}^{(k+1)}}{f_{i^*}^{(k)}} \geqslant \frac{m_{i^*i^*}^{(k)}}{\sum_{l=1}^{M} f_l^{(k)} g_l / g_{i^*}} > \frac{1 - (1 - f_{i^*}^{(k)})(1 - g_{i^{**}} / g_{i^*})}{\sum_{l=1}^{M} f_l^{(k)} g_l / g_{i^*}} =$$

$$\frac{f_{i^*}^{(k)} + (1 - f_{i^*}^{(k)}) g_{i^{**}} / g_{i^*}}{f_{i^*}^{(k)} + \sum_{l\neq i^*}^{M} f_l^{(k)} g_l / g_{i^*}} =$$

$$\frac{f_{i^*}^{(k)} + \sum_{l\neq i^*}^{M} f_l^{(k)} g_{i^{**}} / g_{i^*}}{f_{i^*}^{(k)} + \sum_{l\neq i^*}^{M} f_l^{(k)} g_l / g_{i^*}} \geqslant 1 \tag{13.4.27}$$

从而

$$f_{i^*}^{(k+1)} > f_{i^*}^{(k)} \quad (k = 0, 1, 2, \cdots) \tag{13.4.28}$$

即$\{f_{i^*}^{(k)}\}_{k=0}^{\infty}$为单调上升序列. 注意到:

$$\sup\{f_{i^*}^{(k)}\} = 1 \tag{13.4.29}$$

因此有$\forall k, f_{i^*}^{(k)} \leqslant 1$，以及$\forall \varepsilon > 0$，存在自然数 K 使得 $f_{i^*}^{(K)} \in \{f_{i^*}^{(k)}\}$，且

$$1 - \frac{\varepsilon}{2} \leqslant f_{i^*}^{(K)} \tag{13.4.30}$$

由此及序列$\{f_{i^*}^{(k)}\}_{k=0}^{\infty}$的单调上升性就可推出：$\forall \varepsilon > 0$，当 $k > K$ 时有：

$$1 - \frac{\varepsilon}{2} \leqslant f_{i^*}^{(K)} < f_{i^*}^{(k)} \leqslant 1 \tag{13.4.31}$$

从而

$$1-\varepsilon<1-\frac{\varepsilon}{2}\leqslant f_{i^*}^{(K)}<f_{i^*}^{(k)}\leqslant 1<1+\varepsilon \tag{13.4.32}$$

即

$$|f_{i^*}^{(k)}-1|<\varepsilon \tag{13.4.33}$$

故

$$\lim_{k\to\infty} f_{i^*}^{(k)}=1 \tag{13.4.34}$$

证毕.

遗传算法的选择操作有将群体集中于具有较高适应度区域的能力.而变异操作与之相反，它有增大群体多样性，使群体具有探索可能包含较好的解的新区域的能力.它对于加快搜索速度，确保种群不至于陷入局部极小具有重要的意义，但它无疑要影响群体演变过程的收敛.人们在遗传算法的各种实践中，已经对变异操作提出了许多改进技术，如有指导的变异和各种自适应变异方法等，以兼顾变异操作的得失，但缺乏相应的理论分析.本节的理论分析结果表明，当变异概率随群体状态演变过程的变化变得充分小时，可以保证无杂交操作的遗传算法的演变过程收敛到全局最优状态.这一结论在一定程度上为自适应调节变异操作，以保证遗传算法收敛到全局最优解提供了理论依据.

第十四章　用遗传算法解决组合优化问题

遗传算法重要的研究与应用领域之一是组合优化(combinational optimization)问题.所谓组合优化是指在离散的、有限的数学结构上,寻找满足给定约束条件的最优解.组合优化问题通常具有较多的局部极值点,往往是不可微的、不连续的、多维的、有约束条件的、高度非线性的 NP 完全问题,因此,精确的求解组合优化问题的全局最优解一般是不可能的.遗传算法作为一种模拟进化过程的随机搜索优化方法,在组合优化领域得到了广泛的研究与应用,并在解决许多典型组合优化问题中显示了良好的性能,取得了良好的效果.本章着重介绍遗传算法用于函数优化的若干问题以及遗传算法关于解决邮递员路径问题(TSP)的应用.

§14.1　函数优化

遗传算法应用于纯数学函数优化问题,可以简化遗传算法的工作环境和优化标准,有利于分析和研究遗传算法的基本性能特征.

De Jong 在将遗传算法应用于纯数学函数的优化问题方面,作了许多很有意义的工作.他将 Holland 的模式定理和自己的计算实践结合起来,对基本遗传算法和一些改进的遗传算法的搜索性能进行了较为深入的试验分析,得出了一些很有价值的结论,对遗传算法的发展产生了意要的影响.本节简要介绍一下 De Jong 在函数优化方面的研究工作.

De Jong 精选了五个纯数学函数,这些函数的形式及定义域如表 14.1 所示,经转化得到二维函数特性分别如图 14.1～14.5 所示.

表 14.1　De Jong 所选的五个测试函数

序号	函数	定义域
1	$f_1(x_i)=\sum_{i=1}^{3} x_i^2$	$\lvert x_i\rvert \leqslant 5.12$
2	$f_2(x_i)=100(x_1^2-x_2)^2+(1-x_1)^2$	$\lvert x_i\rvert \leqslant 2.048$
3	$f_3(x_i)=\sum_{i=1}^{5} \text{integer}(x_i)$	$\lvert x_i\rvert \leqslant 5.12$
4	$f_4(x_i)=\sum_{i=1}^{30} i x_i^4+\text{Gauss}(0,1)$	$\lvert x_i\rvert \leqslant 1.28$
5	$f_5(x_i)=0.002+\sum_{j=1}^{25} \frac{1}{j+\sum_{j=1}^{2}(x_i-a_{ij})^2}$	$\lvert x_i\rvert \leqslant 65.536$

表 14.1 中的 5 个函数在数学特征方面的差异主要表理在以下几个方面:

①连续性;②凹凸性;③单峰与多峰函数;④二次函数与非二次函数;⑤低维与高维函数;⑥确定函数与随机函数.

例如,函数 $f_1(x_i)$是一个三维的、连续的、单极值二次函数,函数 $f_2(x_i)$是一个凹函数,

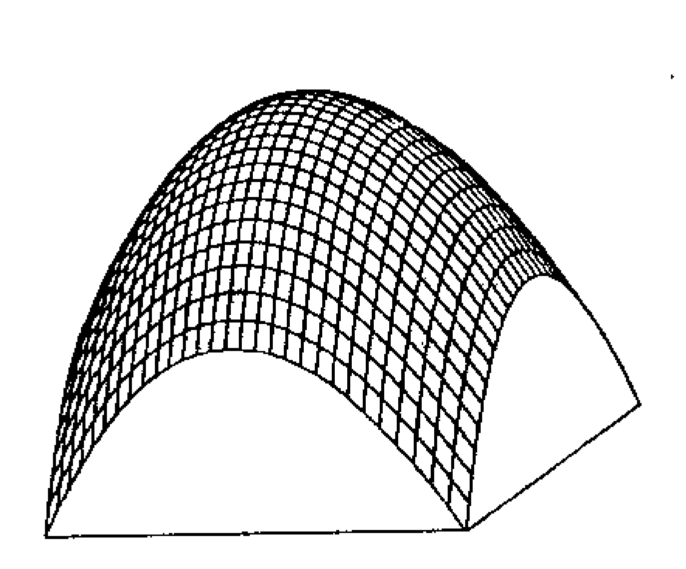


图 14.1 函数 f_1 的几何特性

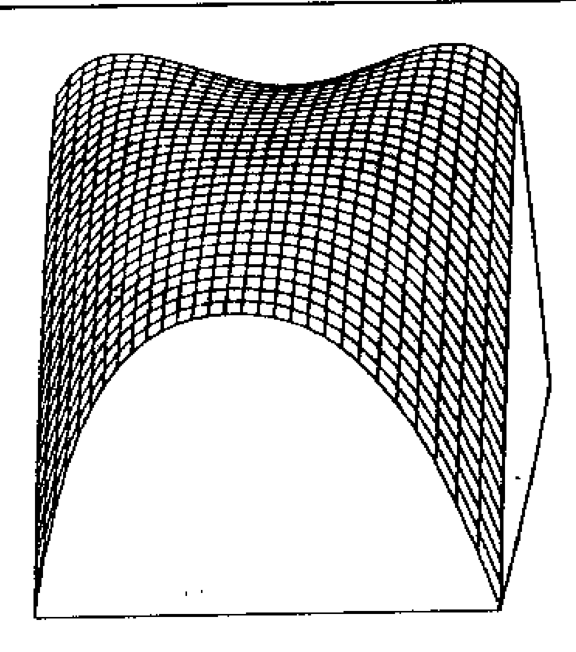


图 14.2 函数 f_2 的几何特性

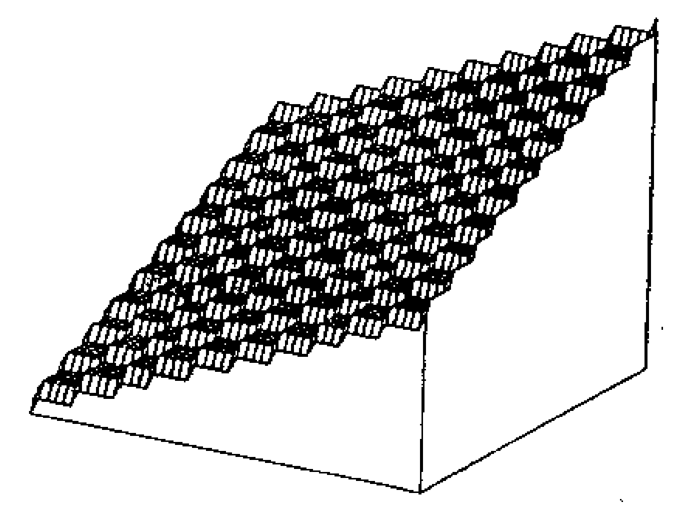


图 14.3 函数 f_3 的几何特性

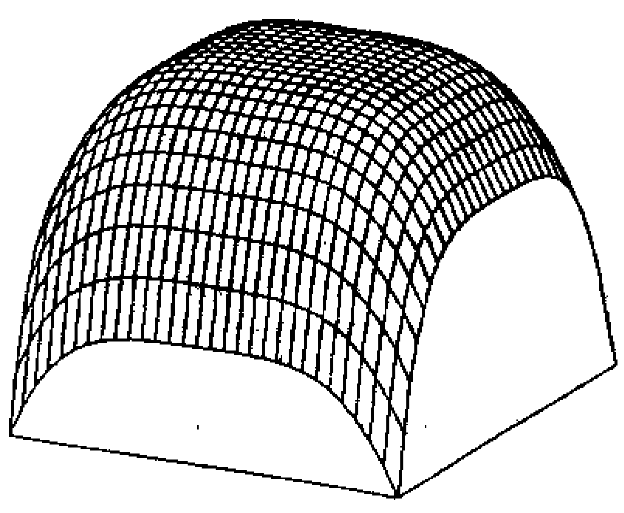


图 14.4 函数 f_4 的几何特性

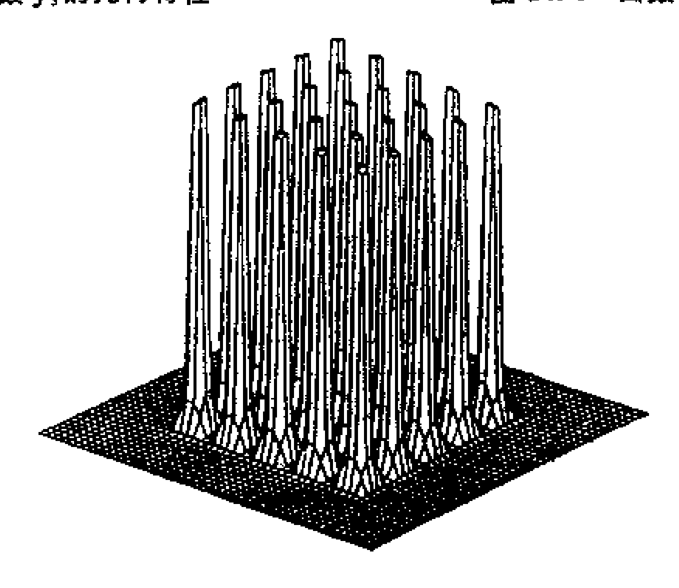


图 14.5 函数 f_5 的几何特性

函数 $f_3(x_i)$具有不连续性，函数 $f_4(x_i)$具有随机特性，函数 $f_5(x_i)$具有多峰特性等.

为了度量不同的遗传算法的性能，De Jong 提出了两个用于定量分析遗传算法的测度，其定义分别如下：

定义 14.1 设 $x_e(s)$为环境 e 下策略 s 的在线性能，$f_e(t)$为时刻 t 或第 t 代中相应于环境 e 的平均适应度函数，则 $x_e(s)$可以表示成

$$x_e(s) = \frac{1}{T}\sum_{t=1}^{T} f_e(t) \tag{14.1.1}$$

上式表明，在线性能可以用从第一代到当前各代的优化进程的平均值来表示，它主要用来测量算法的动态性能.

定义 14.2 设 $x_e^*(s)$为环境 e 下策略 s 的离线性能，则有

$$x_e^*(s) = \frac{1}{T}\sum_{t=1}^{T} f_e^*(t) \tag{14.1.2}$$

其中 $f_e^*(t)=\text{best}\{f_e(1), f_e(2), \cdots, f_e(t)\}$.

上式表明，离线性能是特定时刻最佳性能的累积平均，它主要用来测量算法的收敛性. 依据这两种性能评价标准，针对表 14.1 中的五个基本函数所构成的测试环境. De Jong 探讨和研究了基本 GA 及其具体形式的搜索优化性能.

为便于叙述，将基本遗传算法记做策略 R_1. De Jong 在 R_1策略中考虑了下述相互制约的 4 个参数：

①群体规模 n；②交叉概率 P_c；③变异概率 P_m；④代沟 G，其中 G=1 表示非重叠群体，0<G<1 表示重叠群体.

De Jong 首先考察了在采用策略 R_1时单个参数发生变化的情况，然后考察了它们的有限组合的情况. 关于函数 f_1的群体规模试验表明，较大规模的群体能导致较好的离线性能(收敛性)，但较大群体的初始在线特性较差，而较小群体则显示了较好的初始在线特性. 增大变异概率可以保持群体的多样性，但其代价是降低离线和在线性能. 关于代沟的研究，De Jong 建议非重叠的模型是较好的，离线性能较优.

为了改进基本遗传算法的性能，De Jong 考察了策略 R_1的五种如下变形：

R_2：最佳个体保留模型

R_3：期望值模型

R_4：最佳期望值保留模型

R_5：排挤因子模型

R_6：广义交叉(多点交叉)模型

De Jong 的主要实验结果与分析如下：就 R_2而言，对于单峰函数，最佳保留策略大大改进了在线和离线性能，但在多峰函数 f_5中，最佳保留策略降低了两种性能，这意味着最佳保留策略改进了局部搜索性能，却损失了全局搜索性能. 关于 R_3的实验表明，在离线和在线性能上，R_3明显优于 R_1. 在 5 个函数的环境中 R_3的整体搜索性能优于 R_2和 R_1. 此外，R_3减少了随机误差，能够容许交叉概率的适当增加. 在策略 R_3中，对于单峰函数 $f_1 \sim f_4$，可以观察到相当可观的改进. 而对于多峰函数 f_5，其性能仅低于 R_3. 策略 R_5对于多峰函数的优化是十分有意义的. 策略 R_3的观察表明，多点交叉随着交叉点的增加，降低了在线和离线性能.

§14.2 基于遗传法的 Rosenbrock 函数优化问题

无约束的函数优化是应用范围很广的一类优化问题，随着实际需要的日益增加，对这类问题的研究日见深入. 迄今为止，人们已经提出许多无约束最优化方法，如使用导数的梯度法（最速下降法）、牛顿法、共轭梯度法和变尺度法等，以及不使用导数的各种搜索法.

在函数极小化问题中，目标函数等高面内经常出现（至少在局部范围内）"超山谷"，对于具有两个独立变量的目标函数，在由目标函数等值线绘出的曲线图中则表现为"山谷". 对任何一个成功的最优化方法的主要要求之一，是具有能够在局部区域中沿着窄的山谷迅速移向目标函数极小值的能力，即一个好的算法应给出一个沿山谷有较大分量的搜索方向. 由 Rosenbrock 设计的 Rosenbrock 函数是考察优化方法是否具有这一能力的一个典型例子. 理论与数值研究结果表明，人们广泛采用的、尤其是在远离极小点时颇具优势的最速下降法对于极小化 Rosenbrock 函数是不成功的. 有鉴于此，下面利用遗传法来解决这一问题，并考察了一些改进的遗传算法对子该问题搜索速度的影响，得到了适于解决此类问题的合理的遗传操作，从面为有效地解决最速下降法所不能实现的某一类函数优化问题提供了一种新途径.

需要指出的是，上节所述 De Jong 所选用的函数 $f_2(x_1,x_2)$ 就是 Rosenbrock 函数，这里对其所进行的数值实验结果表明，有些结论是与 De Jong 的结论不一致的，这可能是由于所采用的函数定义域远大子 De Jong 所采用的定义域之缘故.

一、最速下降法用于极小化 Rosenbrock 函数的讨论

首先概述利用最速下降法极小化目标函数 $f(\boldsymbol{x})$ 的问题，其中 $\boldsymbol{x}=(x_1,\cdots,x_n)^{\mathrm{T}}$ 为 n 维矢量，在任一点 $\boldsymbol{x}$ 处，目标函数 $f(\boldsymbol{x})$ 的梯度是 $f(\boldsymbol{x})$ 最大局部增长方向上的一个矢量，于是 $f(\boldsymbol{x})$ 的梯度的反方向即为最速下降方向. 因为在 $\boldsymbol{x}$ 点处，$f(\boldsymbol{x})$ 的负梯度指向 $f(\boldsymbol{x})$ 关于 $\boldsymbol{x}$ 的每个分量减小最快的方向，并与 $f(\boldsymbol{x})$ 在 $\boldsymbol{x}$ 处的等值线正交.

在最速下降法的第 k 步，由一点 $\boldsymbol{x}^{(k)}$ 到另一点 $\boldsymbol{x}^{(k+1)}$ 的移动可由公式

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \Delta\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k)} + \lambda^{(k)}\boldsymbol{S}^{(k)} \tag{14.2.1}$$

给出，其中

$$\boldsymbol{S}^{(k)} = -\frac{\nabla f(\boldsymbol{x}^{(k)})}{\|\nabla f(\boldsymbol{x}^{(k)})\|} \tag{14.2.2}$$

为 $f(\boldsymbol{x})$ 在 $\boldsymbol{x}^{(k)}$ 处的单位负梯度，即最速下降方向，$\lambda^{(k)}$ 为待定因于，由公式

$$\frac{\mathrm{d}f(\boldsymbol{x}^{(k)}+\lambda^{(k)}\boldsymbol{S}^{(k)})}{\mathrm{d}\lambda^{(k)}} = 0 \tag{14.2.3}$$

决定.

下面分析利用最速下降法极小化 Rosenbrock 函数的过程中所遇到的问题. Rosenbrock 函数定义为

$$f(\boldsymbol{x}) = 100(x_2-x_1^2)^2 + (1-x_1)^2 \tag{14.2.4}$$

它的两条等值线（$f(\boldsymbol{x})=4$ 和 $f(\boldsymbol{x})=8$）如图 14.6 所示，从几何上看，$f(\boldsymbol{x})$ 可以解释为一个缓慢地向下弯曲的"山谷"，以 $\boldsymbol{x}^*=(1,1)^{\mathrm{T}}$ 为其最低点，在该点有 $f(\boldsymbol{x}^*)=0$.

由于 Rosenbrock 函数的响应曲面上有一条弯曲两又狭长的山谷，而在山谷中的点的最速

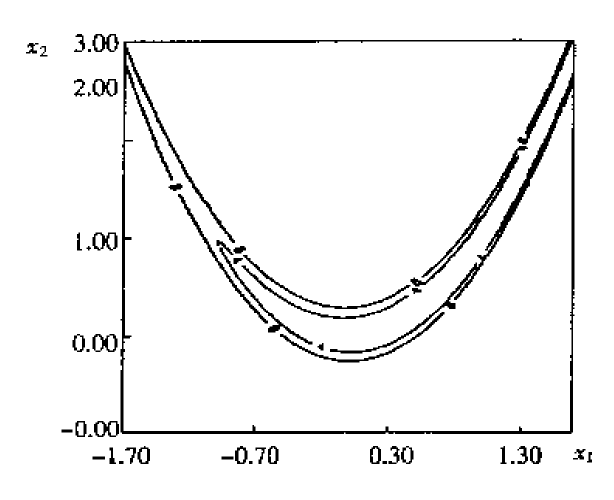


图 14.6　Rosenbrock 函数的等值线图

下降方向几乎与通向 $f(x)$ 极小值的最好方向垂直，所以在山谷中的曲线周围，最速下降法收敛得极慢，甚至在合理的时间内完全不收敛.

最速下降法是无约束优化问题中经常采用的方法之一，尤其在远离极小点时该方法是颇具优势的，但在处理目标函数响应曲面上具有狭长山谷的函数优化问题中，最速下降法是不适用的. 为此，下面采用遗传算法来解决最速下降法所不能实现的这类函数优化问题.

二、若干改进的遗传算法基本操作

由于 GA 在运行过程中需要较长的搜索时间，妨碍着它付诸实际应用，因此改进 GA 的基本操作一直是人们关注的课题，这方面的研究已经取得了相当多的成果，并且针对各种不同类型的问题，达到了提高 GA 运行效率的目的. 本节简要介绍在极小化 Rosenbrock 函数时采用的某些改进的基本操作.

1. 线性选择

线性选择是根据群体 $X(t)$ 中各染色体的适应度，按照一定的概率选出一定数量的染色体，并将它们拷贝到配对库 T 中. 设 $X(t)$ 中和 T 中染色体数目各为 n，则配对库将由 $X(t)$ 中染色体 s_i 的 n_i 个拷贝构成，其中 n_i 极据 R_i 的整数部分以及小数部分的适当选取得出.

$$R_i = \frac{F_i}{\sum_{j=1}^{n} F_j} n \tag{14.2.5}$$

n_i 的选取满足

$$\sum_{i=1}^{n} n_i = n \tag{14.2.6}$$

2. 非线性选择

在产生配对库时引入某些非线性函数，如采用 S-型函数，

$$S(F_i) = \begin{cases} 0, & \text{当 } F_i \leqslant F_{mm} \text{ 时} \\ \dfrac{2(F_i - F_{min})^2}{(F_{max} - F_{min})^2}, & \text{当 } F_{min} \leqslant F_i \leqslant (F_{min} + F_{max})/2 \text{ 时} \\ 1 - \dfrac{2F_{max} - F_i)^2}{(F_{max} - F_{min})^2}, & \text{当} (F_{min} + F_{max})/2 \leqslant F_i \leqslant F_{max} \text{ 时} \\ 1, & \text{当 } F_i \leqslant F_{max} \text{ 时} \end{cases} \tag{14.2.7}$$

其中 F_{max} 和 F_{min} 分别为预先估计的染色体的最大的和最小的适应度值，配对库中染色体 s_i 的

拷贝数目 n_i 根据

$$R_i = \frac{S(F_i)F_i}{\sum_{j=1}^{n} S(F_j)F_j} n \qquad (14.2.8)$$

经适当的舍入计算得出，n_i 的选取满足(14.2.6)式.

3. 单点交叉

单点交叉是随机选择一位置 $r(1 \leqslant r < L)$，交换两个父本染色体的位置 r 右边的部分，产生两个子代染色体.

4. 多点交叉

多点交叉是把长度为 L 的染色体划分为 q 个长度为 l 的子串 $s_i^l(i=1,2,\cdots,q)$产生 q 个在$[1,l-1]$区间内的随机数 r_i^l，将 r_i^l作为子串 s_i^l的交叉点，对每个子串实行单点交叉.

5. 一致交叉

对子被选作父本的染色体随机产生一个同样长度的二进制串，对应于该串中出现“1”的位置，两个父本交换相应的基因.

6. 有指导的变异

设 s_t^* 为第 t 代诸染色体中最适合者，F_t^* 为相应的适应度值，设 $x_t^* = (x_{t1}, x_{t2}, \cdots, x_{tL})^T$ 为 s_t^* 的解码，若在第 $r(r>t)$代最适合的染色体为 s_r^*，相应的适应度值为 F_r^* ($F_r^* > F_t^*$)，则变异操作产生的一个新的点由下式决定

$$x^* = x_r^* + \alpha(1 - t/t_{\max}) g(x_t^*, x_r^*) \qquad (14.2.9)$$

其中 $\alpha(0<\alpha<1)$为常数因子，r 代表在第 t 代后最适应的染色体达到 $F_r^* > F_t^*$ 的第一代，$t_{\max}$ 为迭代的最大次数，$g(x_t^*, x_r^*)$为加速函数，可以是 $F(x)$的梯度，若 $F(x)$处处可微，则 $g(x_t^*, x_r^*)$可取为：

$$g(x_t^*, x_r^*) = F'(x_r^*) \qquad (14.2.10)$$

三、模拟实验结果与分析

针对常规遗传操作和某些改进的遗传操作，对子极小化 Rosenbrock 函数问题进行了较多的模拟实验和分析对比，除了成功地利用遗传算法尝试以外，还考察了一些改进的遗传算法对于该问题搜索速度的影响，得到了适子此问题求解的遗传操作方式，从而为解决最速下降法所不能实现的某一类函数优化问题提供了一种新的、有效的途径.

首先将 x_1, x_2编码为$\{0,1\}$上的位串，由 $q=2$ 个子串构成，每个子串长度为 $l=16$，子串中有一个符号位，四个整数位，其余位表示小数部分.群体内染色体数目为 $n=24$，每次选取 2 个染色体作父本，变异率取为 0.1，适应度函数取为

$$F(x) = \frac{1}{1+f(x)} \qquad (14.2.11)$$

其中 $f(x)$由(14.2.4)式定义.

1. 选择操作

在父本选择中，比较了随机选择、线性选择与非线性选择三种选择操作，实验结果表明，对于 Rosenbrock 函数极小化问题，随机选择是最理想的.尽管线性选择与非线性选择的出发点是好的，它们都可以使得那些适应度值较高的染色体在群体中拥有较多的子代，并且它们已经被证明在处理某些优化问题时是卓有成效的，但在极小化 Rosenbrock 函数的过程中，这两种方法并不能有效

地加快搜索速度,反而会导致在有效时间内达不到理想的结果.在模拟实验中观察到,执行几百代运算后,种群中的染色体都相差无几,适应度值也非常接近.然而整个种群却距离最优点还相差甚远,此后在较长时间内整个种群基本上不再发生变化.

通过分析 Rosenbrock 函数的特殊性,可以得到线性选择与非线性选择失效的原因.由于 Rosenbrock 函数在山谷内的一个很小范围内(与谷底轴向相垂直的方向上)函数值变化很大,因此初始种群中适应度值大的染色体很可能与其他的染色体的适应度值相差若干个数量级.这样,经过线性或非线性选择,该染色体被选中的概率特别大,甚至在较长时间内几乎都是它自身的相互交叉.这样,就使得相差无几的染色体迅速扩张,充满整个种群.它所导致的后果是:①整个种群的适应度值较为接近,使得线性与非线性选择失去作用;②在以后的交叉进程中,相同基因交叉的概率增大;③交叉后所产生的于代与种群中已有的染色体相同的概率增大.这些结果将导致搜索陷入局部极小或未成熟收敛.

鉴于线性与非线性选择不适于极小化 Rosenbrock 函数问题,采用随机选择操作.这种选择操作虽然不能使得整个种群的平均适应度迅速提高,但它可以保证整个配对库中有较丰富的基因供交叉使用,从面避免陷入局部极小或未成熟收敛.

2. 交叉操作

在交叉操作中,比较了单点交叉、多点交叉与均匀交叉三种模式.实验结果表明,对于 Rosenbrock 函数极小化问题,单点交叉并不能在短时间内达到良好的效果,而且有可能在适应度值很低的情况下过早收敛而达到局部极小.这是由于在山谷中欲达到极小点只能沿斜线运动,而单点交叉每次只能改变 x_1 值或 x_2 值的缘故.多点交叉与均匀交叉则完全避免了这一限制,它们都是同时改变 x_1 值和 x_2 值.因此这两种交叉操作可以明显地提高搜索速度并避免陷入局部极小.较多的模拟结果表明,对于 Rosenbrock 函数极小化问题,均匀交叉是最理想的,图 14.7 给出了对于 10 个不同的初始种群,根据分别采用单点交叉、多点交叉与均匀交叉得到的适应度的平均值绘出的适应度随迭代次数变化的比较曲线.表 14.2 给出了对于 20 个不同的初始种群采用多点交叉与均匀交叉操作使适应度值分别达到 0.9 和 0.9999 时所需的迭代次数.

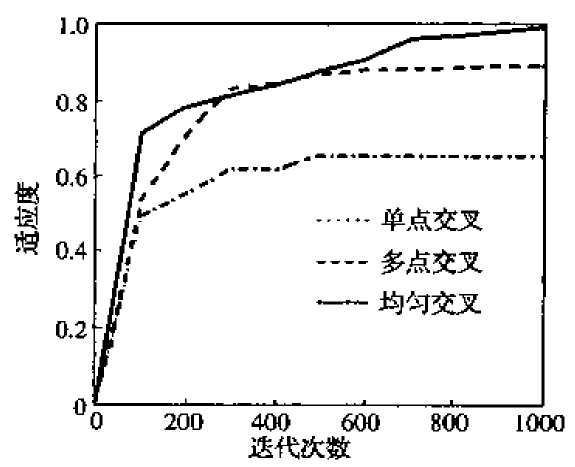


图 14.7 单点、多点与均匀交叉下适应度随迭代次数变化的比较曲线

3. 变异操作

变异是遗传算法的重要操作之一,它既可以产生种群中原本没有的较优基因,也可以恢复在先前的迭代中遗失的较优基因.目前已经提出许多改进技术,如有指导的变异和各种自适应

变异等，它们都针对不同的问题较常规的变异操作具有明显的优越性.

表 14.2 多点与均匀交叉迭代次数比较

	适应度达到 0.9		适应度达到 0.9999	
序号	多点交叉	均匀交叉	多点交叉	均匀交叉
1	345	823	743	1979
2	187	180	2314	1459
3	3139	261	6394	1409
4	258	98	703	527
5	109	409	2563	4436
6	1529	81	1991	2629
7	9776	690	11895	2501
8	150	368	1981	1349
10	106	282	3966	1345
11	65	136	531	1423
12	238	684	1165	4986
13	110	159	1013	719
14	93	393	2392	2811
15	810	520	13901	2760
16	3752	395	5742	5608
17	184	74	3014	1066
18	923	369	2815	926
19	810	173	15345	727
20	314	147	1181	1482

下面分别将常规的变异操作和有指导的变异操作两种方式用于 Rosenbrock 函数极小化问题，并比较和分析了优化结果. 图 14.8 给出了 $\alpha=0, 0.001, 0.002, 0.009$ 时适应度随迭代次数变化的比较曲线，其中 $\alpha=0$ 代表常规的变异操作. 表 14.3 给出了对于 10 个不同的初始种群，α 取不同值并分别迭代到 950 次时所达到的适应度平均值. 由这些结果可以看出，对于

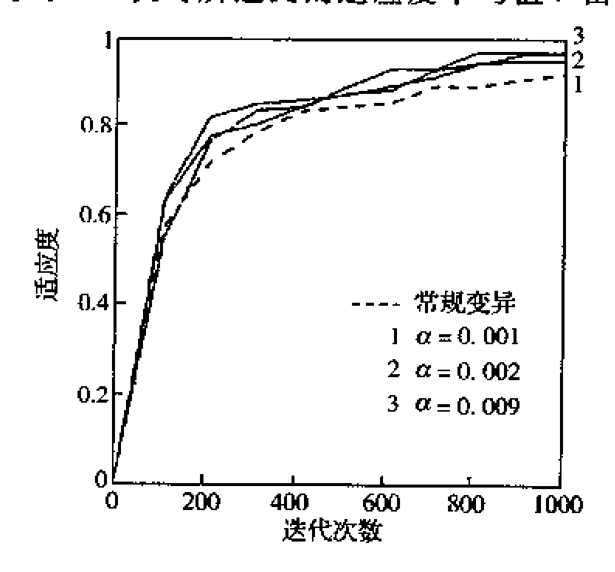


图 14.8 常规变异与有指导变异下适应度随迭代次数变化的比较曲线

Rosenbrock 函数极小化问题，有指导的变异操作的引入，可以稍许改进优化结果. 分析

(14.2.9)式和(14.2.10)式可知，有指导的变异相当于在变异操作中加入了一些最速下降技术，而如前面所述，最速下降法本身是不宜于解决 Rosenbrock 函数极小化问题的.

表 14.3　常规变异与有指导变异在相同迭代次数下达到的适应度值比较

α	0	0.001	0.002	0.006	0.009	0.012
适应度	0.9171	0.9486	0.9637	0.9629	0.9713	0.9599
α	0.04	0.08	0.1	0.3	0.6	0.9
适应度	0.9670	0.9649	0.9691	0.9655	0.9665	0.9665

§14.3　邮递员路径问题(TSP)

TSP 问题属于 NP 完全问题，若用穷举搜索算法，则须考虑所有可能的情况，找出所有的路径，再对其进行比较，来找到最佳路径. 这种方法随着城市数 n 的上升，算法时间随 n 按指数规律增长，即存在所谓的指数爆炸(也称组合爆炸)问题.

事实上，在 n 个城市的 TSP 问题中，一条有效的路径可以看成 n 个城市的一种排列. n 个城市有 $n!$ 种排列. 注意到两个顺序完全相反的方案其行程相同，而对于一种排列从哪个城市出发都可以，所以有效路径的方案数目为 $R_n=\frac{n!}{2n}$. 例如，$R_4=3$，$R_5=12$，$R_6=120$，$R_{10}=181440$，$R_{30}\approx4.4\times10^{30}$，$R_{60}\approx6.93\times10^{78}$，$R_{100}\approx4.67\times10^{155}$. 可见路径总数随 n 增大急剧增长，当城市数目增加，计算量增加到无法进行的地步.

近年来，基于遗传算法求解 TSP 问题的研究十分活跃. 在遗传算法研究中，TSP 问题经常被用来评价不同的遗传操作及改进的遗传算法的性能. 之所以如此，主要有以下几个方面的原因：

(1) TSP 问题是一个典型的，易于描述却难以处理的 NP 完全问题，有效地解决 TSP 问题有重要的理论价值.

(2) TSP 问题是许多领域内出现的多种复杂问题的集中概括和简化形式. 因此，快速、有效地解决 TSP 问题有很高的实际应用价值.

(3) 由于 TSP 问题的典型性，它已成为各种启发式的搜索、优化算法的间接比较标准. 面遗传算法就其本质来说，主要是处理复杂问题的一种鲁棒性强的启发式随机搜索算法. 遗传算法在 TSP 问题求解方面的应用研究，对于构造合适的遗传算法框架、建立有效的遗传操作具有重要意义.

一、编码与适应度函数

在求解 TSP 问题的各种遗传算法中，多采用以遍历城市的次序排列进行编码的方法. 如码串 123456789 表示自城市 C_1 开始，依次经城市 C_2，C_3，…，C_9 最后返回城市 C_1 的遍历路径. 这是一种针对 TSP 问题的最自然的编码方式，不过这一编码方案引起了交叉操作的困难，所以应在交叉操作时设计特殊的适合于处理 TSP 问题的交叉策略.

TSP 问题是寻找 n 个城市的排列，使有效路径长度

$$T_d=\sum_{i=1}^{n-1}d_{i,i+1}+d_{n,1}\tag{14.3.1}$$

取最小值，式中 d_{ij} 表示城市 C_i 到城市 C_j 的距离.

在利用遗传算法求解 TSP 问题中，适应度函数常取路径长度的倒数，即 $f=\frac{1}{T_d}$. 若考虑 TSP 问题的约束条件(每个城市经过且只经过一次)，则适应度函数可表示为：

$$f=\frac{1}{T_d+\alpha N_t} \tag{14.3.2}$$

其中 N_t 为对于 TSP 路径不合法的度量(如取 N_t 为未遍历的城市的个数)，α 为惩罚系数.

二、交叉策略

对于 TSP 问题，基于上述顺序编码，若用单点交叉或多点交叉策略，必然以极大的概率导致未能完全遍历所有城市的非法路径，如 TSP 问题的两个父本路径为

1234 | 56789

9876 | 54321

若用单点交叉，且交叉点随机选为 4，则交叉后产生的两个后代为

123454321

987656789

显然，这两个子路径均没能遍历所有 9 个城市，都违反 TSP 问题的约束条件. 针对这一问题，虽然可以采用构造惩罚函数的方法，但实验效果不佳. 理论上的解释是，用惩罚函数法将本已十分复杂的 TSP 问题更加复杂化了. 因为满足 TSP 问题约束条件的可行解空间规模为 $(n-1)!$，而构造惩罚函数的方法，其搜索空间规模变为 $(n-1)^{n-1}$，随着 n 的增大，两者之间的差距是极其惊人的.

解决 TSP 约束问题的一种有效的处理方法是对交叉、变异等遗传操作做适当的修正，使其自动满足 TSP 的约束条件，现介绍常用的几种交叉方法：

1. 部分匹配交叉(Partially Matched Crossover，PMX)法

在 PMX 操作中，先依据均匀随机分布产生两个位串交叉点，定义这两点之间的区域为一匹配区域，并使用位置交换操作交换两个父串的匹配区域. 然后再对两子串匹配区域以外出现的遍历重复，依据匹配区域内的位置映射关系，逐一进行交换.

例如，设两父串及匹配区域为

$A = 91 \mid 456 \mid 7832$

$B = 68 \mid 123 \mid 9547$

首先交换 A 和 B 的两个匹配区域，得到

$A' = 91 \mid 123 \mid 7832$

$B' = 68 \mid 456 \mid 9547$

然后，对于 A'、B' 两子串中匹配区域以外出现的遍历重复进行交换. 对于 A' 有 $1\to4, 2\to5, 3\to6$，于是得

$A'' = 941237865$

同理可得

$B'' = 384569217$

2. 顺序交叉(Order Crossover，简称 OX)法

OX 与 PMX 法类似，也是首先选择一个匹配区域：

$A = 91 \mid 456 \mid 7832$

$$B = 68 \mid 123 \mid 9547$$

然后，根据匹配区域的映射关系，在其区域外的相应位置标记 H，得

$$A' = 9H \mid 456 \mid 78HH$$

$$B' = H8 \mid 123 \mid 9HH7$$

再移动匹配区至起点位置，且在其后预留相等于匹配区域的空间（H 的数目），然后将其余的码按相对次序排列在预留区后面，得到

$$A'' = 456HHH789$$

$$B'' = 123HHH978$$

最后将父串 A,B 的匹配区域相互交换，并放置到 A'',B''的预留区内，即可得到两个子代

$$A''' = 456123789$$

$$B''' = 123456789$$

与 PMX 法比较，OX 法在两父串相同的情况下，也能产生有一定改变的子串，这对于维持群体内个体的多样性具有一定的作用. 还可以设计其他交叉方法，如一种改进的 OX 法如下：

设两父串及匹配区域为

$$A = 91 \mid 456 \mid 7832$$

$$B = 68 \mid 123 \mid 9547$$

首先将 A 的匹配区加到 B 的前面，B 的匹配区加到 A 的前面，

$$A' = 123914567832$$

$$B' = 456681239647$$

然后，在 A'和 B'中分别在匹配区后依次删除与匹配区相同的城市码，得最终子串为

$$A'' = 123945678$$

$$B'' = 456812397$$

三、变异操作

针对 TSP 问题主要的变异技术如下：

1. 逆转变异

在码串中随机选择两个逆转点，然后将这两点内的子串逆向排序. 例如，设串 A 的逆转点为 3,6，则经逆转变异后变为 A'，有

$$A = 123 \mid 456 \mid 789$$

$$A' = 123 \mid 654 \mid 789$$

2. 对换变异

随机选择码串中两点作为对换点，对换其值. 如对于串 A，选择对换点为 3,8，得

$$A = 12\,\underline{3}4567\,\underline{8}9$$

则经对换后变为 A'，

$$A' = 12\,\underline{8}4567\,\underline{3}9$$

3. 插入变异

从码串中随机选择一个码，将此码插入随机选择的插入点中间. 例如对于上述串 A，若取插入码为 6，选取插入点为 3～4 之间，则有

$$A' = 123645789$$

四、求解 TSP 问题的实验结果

在实验中，群体规模定为 100，交叉操作采用上述改进的 OX 法，在 386 微机上的有关实验结果如下：

(1) 当 $n\leqslant 15$ 时，随机样本实验表明，利用遗传算法可 100%搜索到用穷举法求解的最优解.

(2) 当 $15\leqslant n\leqslant 30$ 时，计算结果表明 GA 能收敛到一稳定的"最好解"，实验误差为 0，模拟退火(SA)也可求得相同的"最好解"，但运行时间约为 GA 的 6 倍(此时难以确认最优性).

(3) 对 $n=50$，$n=60$，$n=80$ 及 $n=100$ 的测试结果表明，GA 在求解质量上略优于 SA，优化效率高于 SA，表 14.4 给出了 GA 与 SA 的实验结果比较，表中所列 TSP 路径长度为相对长度，其值由下式给出：

$$T_d'=\frac{T_d}{0.765X\sqrt{n}} \tag{14.3.3}$$

其中，T_d 为实际路径的长度，X 为包含 TSP 所有城市的最小正方形的边长，n 为 TSP 问题的城市数目. 图 14.9 为城市规模 $n=100$ 的 TSP 问题的初始最佳路径，图 14.10 为经 GA 优化得到的最佳路径.

表 14.4　SA 法和 GA 求解 TSP 问题的实验结果

城市数 n	最佳解 SA 法	最佳解 GA 法	时间/s SA	时间/s GA
50	106.33	105.88	540	98
60	105.11	104.22	480	120
80	103.22	101.34	600	150
100	99.11	98.05	1080	360
200	98.67	97.91	3400	660

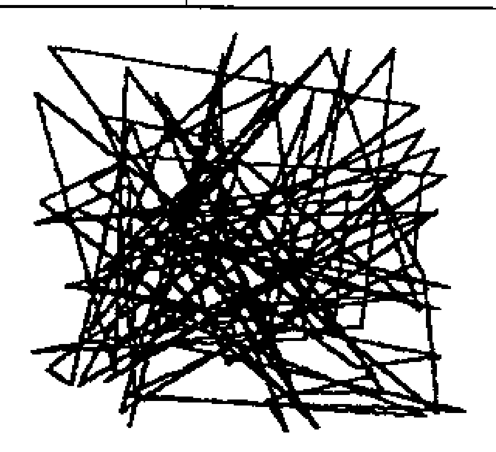


图 14.9　100 城市 TSP 问题的初始最佳路径($G=0$ 代)

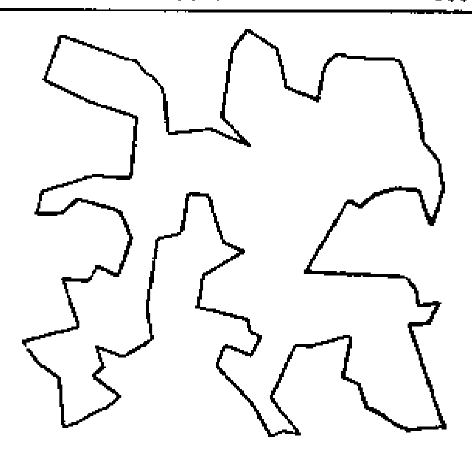


图 14.10　100 城市 TSP 问题经 GA 优化后的最佳路径($G=200$ 代)

第十五章　其他进化算法

进化计算通常分为遗传算法(Genetic Algorithm,GA)、进化策略(Evolution Strategies,ES)、进化规划(Evolutionary Programming,EP)与遗传规划(Genetic Programming,GP)四种典型策略.

§15.1　遗传规划概述

15.1.1　遗传算法的局限性

从前面的讨论可以看到,遗传算法通过对字符串进行操作,可以解决一系列复杂的问题.然而正是由于遗传算法直接对字符串进行操作,因此在遗传算法中对问题的描述将显得至关重要.在遗传算法的理论与应用研究中,人们在感受编码方法的重要性与优越性的同时,也体会到了它对于遗传算法应用范围的限制.

遗传法的主要局限性如下:

1. 不能描述层次化问题

有许多问题,其解答的自然描述往往可以用一种层次化的计算机程序表达,而不是一种定长的字符串形式的表达.例如,常用的函数表达式 $f(x)=A_0+A_1x+A_2x^2+A_3x^3$,事实上可以看成一个具有层次化结构的计算机程序,图 15.1 直观地描述了该层次化结构.

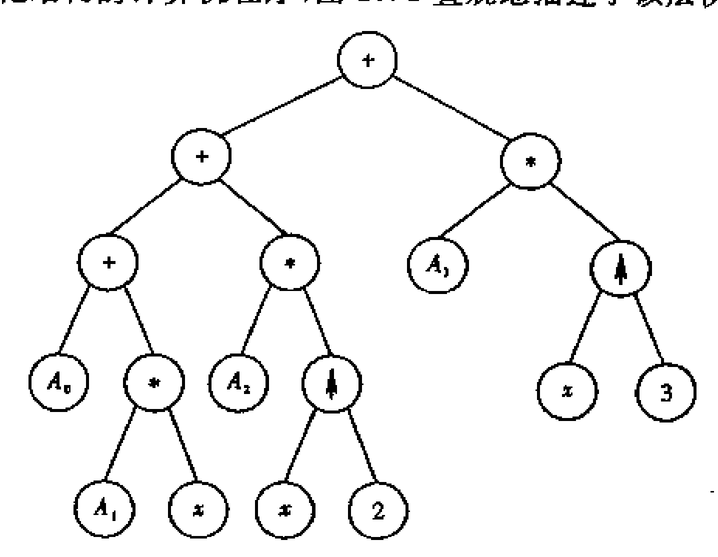


图 15.1　函数表达式 $f(x)$ 的层次化结构

一般情况下,这种层次化的计算机程序的结构和大小在问题获得解决之前往往无法了解.例如,对图 15.2 所示的数据进行曲线拟合时,究竟选下列哪一个函数进行拟合,事先是无法了解的:

$$f(x)=A_0+A_1x+A_2x^2+A_3x^3$$

$$f(x)=A_0\log(A_1x+A_2x^2+A_3x^3)$$

$$f(x)=A+\exp(A_1x)+\log(A_2x^2)+\sin(A_3x^3)$$

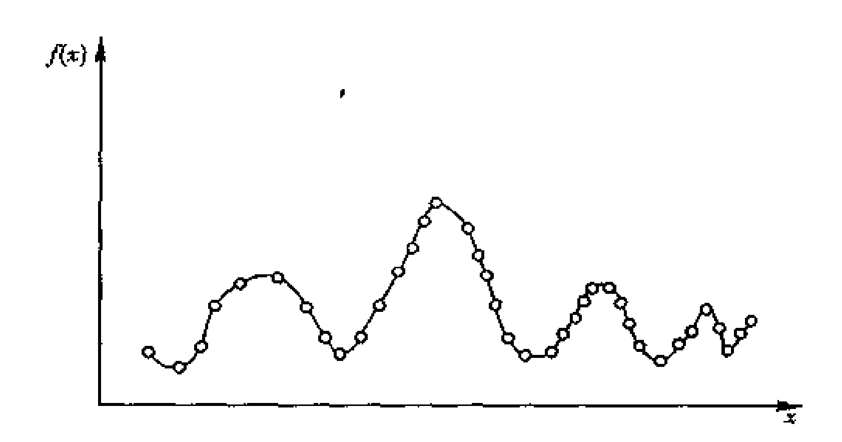


图 15.2 某事件的实测数据

因此，这种程序在随机确定其初态后，应具有根据所在环境的状况修改其结构和大小的能力. 显然，用定长字符串方法来描述上述这种变化的计算机程序是困难的.

2. 不能描述计算机程序

即使计算机程序已经确定，利用定长字符串方法也不能方便地描述这种计算机程序，如样本数据回归、方程求解以及包含有递归、迭代等过程的算法等.

3. 缺少动态可变性

定长的字符串描述方法不具备动态可变性，字符串的长度一旦确定，就很难改变系统的内部状态及系统的行为.

遗传算法中定长字符串的采用，相当于对问题解答的结构和大小的一种预确定，它极大地限制了遗传算法在许多方面，如人工智能、机器学习和符号处理等领域的应用.

15.1.2 遗传规划的提出

鉴于遗传算法的局限性，人们不断探讨对问题进行描述的断方法. 早在 20 世纪 80 年代中期，人们已开始努力改进遗传算法的表达式，例如允许用不定长的字符串或添加一系列 if…then 条件语句等. 为了克服遗传算法在表达方面的局限性，人们提出了遗传规划(Genetic Programming ,GP)，用计算机程序来代替字符串表达所研究的问题. 1989 年，美国斯坦福大学的 Koza J R 提出了遗传规划的新概念，用层次化的计算机程序代替字符串的表达问题. 1992 年，他又出版了专著"*Genetic Programming On the Programming of Computers by Means of Natural Selection*"(《遗传规划——应用自然选择的计算机程序设计》)，该书全面介绍了遗传规划的原理及应用实例. 它表明，作为一种新技术，遗传规划已经与遗传算法并驾齐驱.

Koza J R 被称为遗传规划的奠基人，1994 年，他出版了第二本专著"*Genetic Programming* Ⅱ: *Automatic Discovery of Reusable Programs*"(《遗传规划Ⅱ:可重用程序的自动发现》)，进一步阐明了遗传规划的实质. 同年，Kinnear K E Jr. 主编了"*Advances in Genetic Programming*"(《遗传规划进展》)，汇集了许多研究工作者有关遗传规划的应用技术和经验，这标志着遗传规划已渗透到生命科学、工程技术及社会科学的各个领域中.

15.1.3 遗传规划简介

遗传规划是以计算机程序的形式表达问题，它的结构和大小都是可以变化的，从而便于表达问题的复杂性质.

现以曲线拟合为例，简要说明遗传规划的工作原理. 图 15.3 的曲线表示实验结果，现在要确定实验结果的函数关系 $y=f(x)$.

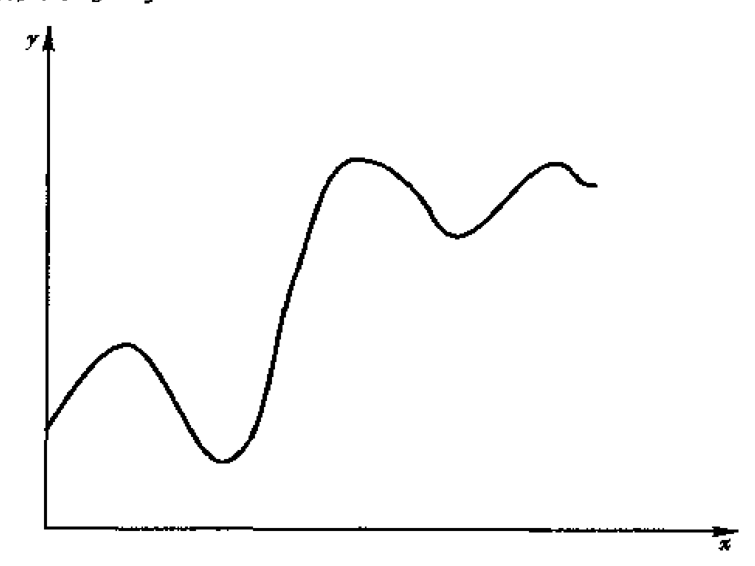


图 15.3　实验结果曲线

遗传规划的工作原理大致如下：

随机地产生初始结构. 设所产生的 $y=f(x)$ 的初始表达式有下述四种：

① $$y = A+Bx$$

② $$y = A+Bx+Cx^2$$

③ $$y = x\sin x$$

④ $$y = Dx\sin x$$

计算适应度. 将不同的 x_i 代入四种表达式中，从而得出一组不同的 y_i^*. 将计算所得的 y_i^* 与实验数据 y_i 相比较，并确定可以衡量表达式优劣的量作为适应度. 假设第 3 种表达式最佳，第 1 种表达式最差.

选择. 根据优胜劣汰的原则，选择第 2 种、第 4 种表达式各一份. 第 3 种表达式二份，淘汰效果最差的第 1 种表达式，于是经选择后的表达式由下述方程组成：

① $$y = x\sin x$$

② $$y = A+Bx+Cx^2$$

③ $$y = x\sin x$$

④ $$y = Dx\sin x$$

交叉. 为了产生新的表达式，要使用交叉操作. 随机地选取交叉对和交叉位置. 假设对第 2、3 种表达式进行交叉，交换位置在第一项，于是新的表达式为：

① $$y = x\sin x$$

② $$y = x+Bx+Cx^2$$

③ $$y = A\sin x$$

④ $$y = Dx\sin x$$

(5) 变异. 在遗传规划中也可采用变异产生新个体. 例如，将表达式中的 $\sin x$ 变为 $\cos x$. 不过，遗传规划中的变异远不及在遗传算法中那样重要.

上述步骤(2)～(5)反复执行，使函数表达式 $y=f(x)$ 不断进化，逐步得出所要求的表达

式.

从上述简单例子可以看出，遗传规划和遗传算法相类似，同样具有选择、交叉、变异等操作，并都利用适应度作为目标函数，一代一代地进化，逐步得出最优的数学表达式.

§15.2　遗传规划基本原理

15.2.1　个体的描述方法

在遗传规划中，首先要解决的问题是如何使用一系列可行的函数对个体进行描述. 遗传规划中可能个体的集合是函数和端点所有可能组合的集合，它们可以从 N_f 个函数的集合

$$F = \{f_1, f_2 \cdots, f_{N_f}\}$$

和 N_t 个端点的集合

$$T = \{a_1, \cdots a_2, \cdots, a_{N_t}\}$$

中递归地构成. 函数集 F 中每个函数 f_i 有 $Z(f_i)(i=1,2,\cdots,N_f)$ 个自变量.

函数集 F 中的函数可以包括：

①算术运算符，如＋、－、＊、/等；

②标准数学函数，如 sin、cos、exp、log 等；

③布尔运算符，如 AND、OR、NOT 等；

④条件表达式，如 if-then-else 等；

⑤可迭代函数，如 do-until 等；

⑥可递归函数；

⑦任何其他可定义的函数.

端点一般要么是变量（如描述系统的输入、传感器、探测器状态变量等），要么是常量. 端点有时也可以是没有显式自变量的函数.

考虑下列函数集

$$F = \{\text{AND}, \text{OR}, \text{NOT}\}$$

和端点集

$$T = \{D0, D1\}$$

其中 $D0$, $D1$ 为布尔变量，也可视为无自变量的函数. 现考虑函数集和端点集的并集 C 为：

$$C = F \cup T = \{\text{AND}, \text{OR}, \text{NOT}, D0, D1\}$$

显然并集 C 中的端点可视为具有 0 个自变量的函数. 于是并集 C 中的函数自变量的个数分别为：2，2，1，0，0.

考虑如下布尔型函数的符号表达式：

$$(\text{NOT}(D0)\ \text{AND NOT}\ (D1))\ \text{OR}\ (D0\ \text{AND}\ D1)$$

图 15.4 描述了上述符号表达式的层次化结构（树）. 树中，5 个内节点为函数集 F 中的函数元素（OR，AND，AND，NOT，NOT）；4 个外节点（叶子）为端点集 T 中的布尔变量（$D0$，$D1$，$D0$，$D1$）；根为符号表达式中的一个函数，即 OR. 该树即为数据结构理论中著名的算法树数据结构、这种算法树数据结构非常便于表达式求值.

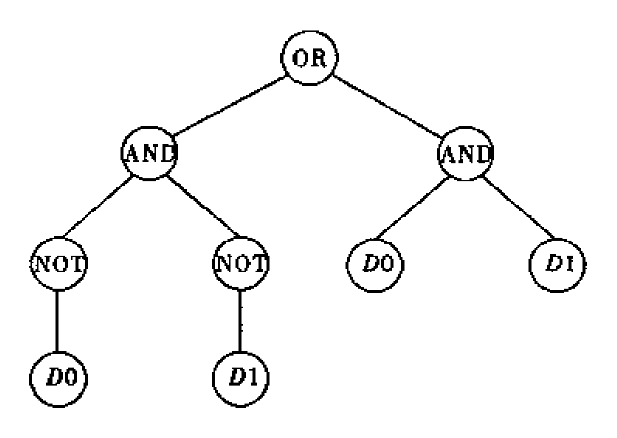


图 15.4 具有有序分支的树

15.2.2 初始群体的生成

初始群体由多个初始个体组成，初始个体是所要解决问题的各种可能的符号表达式(算法树)，它通过随机方法产生.

开始时，在函数集 F 中按均匀分布随机地选出一个函数作为算法树的根节点. 把根节点的选择限制在函数集 F 中是为了生成一个层次化的结构，而不是产生由单个端点构成的退化结构. 图 15.5 描述了函数“+”(具有 2 个自变量)为根节点的算法树，函数“+”是随机地从函数集 F 中选出来的. 一般情况下，从函数集 F 中选出的函数 f，如果有 $Z(f)$ 个自变量，那么就要从节点发出 $Z(f)$ 条线. 然后，对于每条从节点发出的线，从函数集 F 和端点集 T 的并集 C 中用按均匀分布随机地选出的一个元素作为该条线的尾节点. 如果此时选出的是一个函数，则重复上述过程. 例如，若函数“ * ”从并集 C 中被选出，则函数“ * ”将作为非根内节点，如图 15.6 中的点 2 所示. 由于函数“ * ”具有两个自变量，所以从节点 2 发出两条线. 如果从并集 C 中被选出的是端点，它作为从节点发出线的尾节点，则该分支上的树就终止生长. 例如，在图 15.7 中，恰巧端点 A, B, C 分别从并集 C 中被选出，并作为树中各分支的尾节点，这时整个算法树生成过程终止.

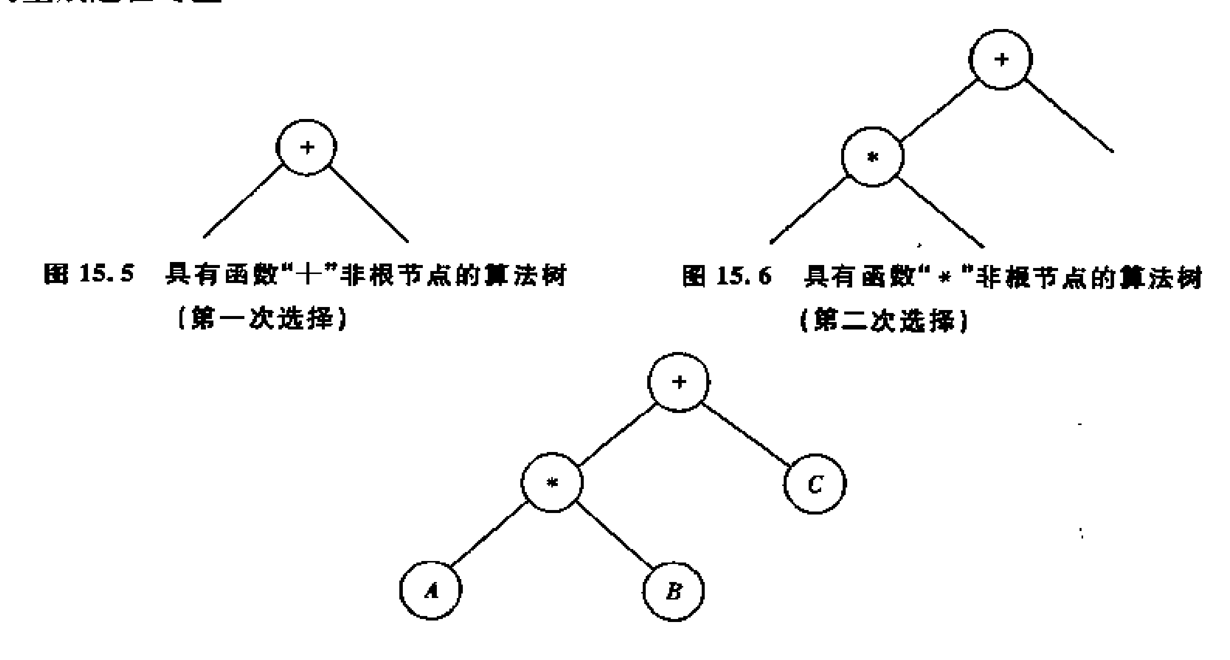


图 15.5 具有函数“+”非根节点的算法树(第一次选择)

图 15.6 具有函数“ * ”非根节点的算法树(第二次选择)

图 15.7 端点 A, B, C 被选出作为叶子的算法树(第三～五次选择)

上述过程从上到下，从左到右不断重复，直到一棵完整的树生成为止.

初始个体的生成可使用不同的方法，从而得到具有不同规模和形状的初始个体. 下面介绍两种产生初始随机树的基本方法，为此首先定义叶子的深度和树的深度. 叶子的深度定义为叶子距树根的层数. 树的深度定义为从树根到终端节点中最大的层数. 例如，图 15.7 的树深度为 3.

1. 完全法

用完全法产生的初始个体，每一叶子的深度都等于给定的树的深度. 其实现方法是，若待定节点的深度小于给定的树的深度，则该节点的选择将限制在函数集内；若待定节点的深度等于给定的树的深度，则该节点仅从端点集内选取.

2. 生长法

用生长法产生的初始个体具有不同的形状，每一叶于的深度不一定都等于给定的树的深度. 其实现的方法是：若待定节点的深度小于给定的树的深度，则该节点的选择将限制在函数集与端点集组成的并集内；若待定节点的深度等于给定的树的深度，则该节点仅从端点集内选取.

15.2.3 适应性度量

在遗传规划中，经常用到以下四种适应性度量.

1. 原始适应度(Raw Fitness)

原始适应度是问题适应性自然描述的一种度量. 在遗传规划中，适应度通常是在一组适应度计算事例上进行度量. 适应度计算事例一般仅是整个搜索空间中一个少量的有限抽样，但同时也要求它们从整体上必须是搜索空间的代表事例，因为它们是对从整个搜索空间中得到的结果进行一般化的基础.

通常把误差定义为原始适应度，也就是说，把一个符号表达式在这些适应度计算事例上返回的值与问题的正确答案之间的距离的和作为其适应度值. 符号表达式返回的值可以是整数值、浮点值、复值、向量值或布尔值.

若符号表达式是整数型或浮点型，那么距离之和表现为符号表达式的返回值与实测值之间的距离绝对值的总和. 在群体中，子代 t 中的某一个体 i 的原始适应度 $r(i,t)$ 定义为：

$$r(i,t) = \sum_{j=1}^{N_c} | S(i,j) - C(j) | \qquad (15.2.1)$$

其中，$S(i,j)$ 为个体 i 在适应度计算事例 j 下的返回值；N_c 为适应度计算事例数；$C(j)$ 为适应度计算事例 j 的实测值或正确值.

若符号表达式是布尔型或字符型，那么距离之和表现为符号表达式的返回值与实测值之间的符号不匹配的个数.

由于原始适应度是问题的自然表达，因此原始适应度的最佳值或越小越好，或越大越好.

2. 标准适应度(Standardized Fitness)

为了把原始适应度最佳值取最小和最大两情形统一起来，下面从原始适应度导出标准适应度 $S(i,t)$. 标准适应度总是表现为数值越小适应性越好.

若在特定问题中，原始适应度越低越好，则标准适应度即可取为原始适应度，即：

$$S(i,t) = r(i,t) \qquad (15.2.2)$$

若在特定问题中，原始适应度越大越好，则标准适应度可定义如下：

$$S(i,t) = r_{max} - r(i,t) \tag{15.2.3}$$

其中，r_{max}为原始适应度所能达到的最大值.

3. 调整适应度(Adjusted Fitness)

子代 t 中个体 i 的调整适应度 $a(i,t)$由标准适应度计算而得，即：

$$a(i,t) = \frac{1}{1+S(i,t)} \tag{15.2.4}$$

因为 $S(i,t)\geqslant 0$，故 $a(i,t)\in[0,1]$，且调整适应度值越大，个体越优良. 当标准适应度接近于 0 时，调整适应度具有扩大标准适应度值微小差别的好处. 这一情形常出现在进化的最后几代中. 随着群体的不断进化，寻优的着重点就要放在区分好的和更好的个体上，但它们之间的差别往往很小. 特别是当接近正确答案时，若标准适应度接近于 0，则调整适度的扩张能力就显得特别有效. 例如，当标准适应度取值介于 0(最佳)和 64(最差)之间时，标准适应度为 64 和 63 的两个较差个体的调整适应度分别为 0.0154，0.0156. 其差为 0.0002，而标准适应度为 4，3 的两个优良个体的调整适应度分别为 0.20，0.25，其差别为 0.05.

4. 归一化适应度(Normalized Fitness)

归一化适应度 $n(i,t)$由调整适应度计算而得，即

$$n(i,t) = \frac{a(i,t)}{\sum_{j=1}^{n} a(j,t)} \tag{15.2.5}$$

其中 n 为群体中的个体数目.

归一化适应度有以下三条性质：

①$n(i,t)\in[0,1]$；

② 适应度值越大，个体越优；

③ $\sum_{i=1}^{n} n(i,t)=1$.

如果个体选择方法基于适应度比例进行，那么归一化适应度可直接作为个体选择的概率.

15.2.4 主要操作

在遗传规划中，有两个主要操作：选择操作和交叉操作.

一、选择

选择操作仅作用于一个父代符号表达式，每执行一次选择只产生一个子代符号表达式. 选择操作分为两步，首先基于适应度值按某种选择方法从当前群体中选择一个符号表达式，然后把它复制到新一代群体中.

基于适应度的选择方法有许多种，其中一种常用的方法是与适应度成比例的选择方法，即赌轮选择，在前面章节所介绍的遗传算法的几种选择方法，在遗传规划中同样适用.

二、交叉

在遗传规划中，交叉是生成新个体的活动. 新个体通过父代个体提供不同组成部分而产生. 交叉时，两个父代个体产生两个于代个体.

1. 交叉操作的实现方法

与选择过程相类似，根据某种选择方法独立地从群体中选出一对父代个体. 一般来说，所

选出的两个父代个体的结构、大小可能都不相同. 交叉时,在每个父代个体的算法树上分别按均匀概率分布随机选择一个交叉点,于是产生一棵以交叉点为根的子树,该子树包括交叉点以下的所有子树. 具有这样特点的一棵子树称为一个交叉段. 有时,一个交叉段仅是一个叶子.

为了生成第一个子代个体,首先将第一个父代个体删掉其交叉段,然后将第二个父代个体的交叉段插入到第一个父代个体的交叉点处,第二个子代个体按对称的方式生成.

例如,考虑如下两个父代符号表达式:

NOT(*D*1) OR (*D*0 AND *D*1)

(*D*1 OR NOT(*D*0) OR (NOT(*D*0) AND NOT(*D*1))

这两个符号表达式可视为如图 15.8 所示的两个算法树.

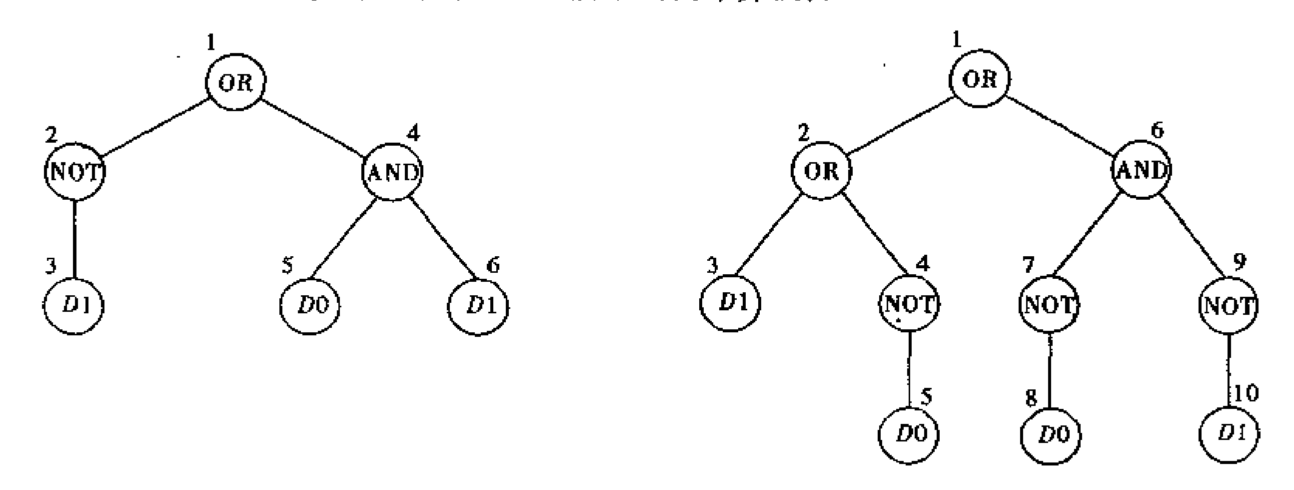


图 15.8　两个父代个体(符号表达式)

假设树的节点按深度优先,从左到右的方式编号. 假定第一棵树的第 2 个节点被随机选定为交叉点,即交叉点为 NOT 函数. 假定第二棵树的第 6 个节点被随机选定为交叉点,即交叉点为 AND 函数. 第一棵树的交叉段是以节点 2 为根的子树,遗留部分称为剩余段,第二棵子树的交叉段是以节点 6 为根的子树. 图 15.9 描述了两个交叉段,图 15.10 描述了交叉后的两个子代个体. 两个子代个体的符号式分别为:

(NOT(*D*0) AND NOT(*D*1)) OR (*D*0 AND *D*1)

(*D*1 OR NOT(*D*0)) OR NOT (*D*1)

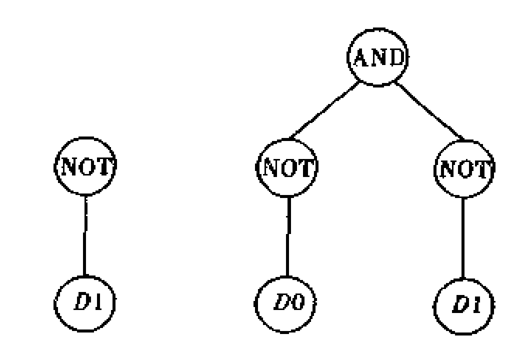


图 15.9　两个由父代个体产生的交叉段

2. 交叉操作的一般特征

由于交叉操作时整个子树被交换,所以不管父代个体和交叉点如何选择,交叉操作所产生的新的子代符号式总是语法上合法的表达式.

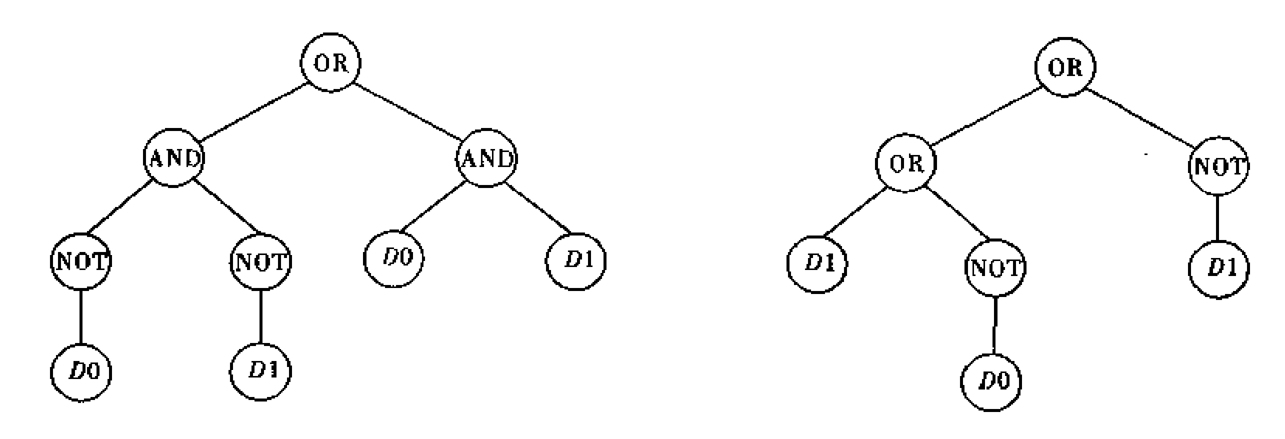


图 15.10 两个由父代个体产生的子代个体

如果一个父代个体的交叉段恰为一个端点(叶子),而另一个父代个体的交叉段为一棵子树,那么在交叉时,第一个父代个体的这个叶子(交叉段)将插在第二个父代个体剩余段的交叉点处,第二个父代个体的子树(交叉段)将插在第一个父代个体剩余段的交叉点处.这样,交叉操作往往会产生深度较大的子代个体.

如果两个父代个体的交叉段均为端点(叶子),那么交叉操作仅表现为两深树之间的叶子交换.这相当于一棵树的叶子发生变异.因此,点变异是交叉操作的一种退化表现.

如果一个父代个体的交叉段恰为整个树(即交叉点为根),而另一个父代个体的交叉段为一棵子树,那么在交叉时,第一个父代个体将整个插在第二个父代个体剩余段的交叉点处,第一个父代个体变为子代个体的一棵子树.这样,往往也会产生深度很大的子代个体.

在交叉操作中,子代个体的大小(一般用树的深度衡量)应有一定的限制.这种限制将防止因少数子代个体太大而花费太多的计算时间.如果两个父代个体交叉后产生一个巨型子代个体,而另一个是正常子代个体,则将后者保留,并任选一个父代个体实施复制代替前者.

如果两个父代个体的交叉点都是根节点,那么交叉操作退化为一种简单的复制操作.

3. 交叉操作的特殊作用

在遗传算法中,如果两个相同的个体进行交叉,那么所产生的子代个体将是相同的.这是因为个体结构是定长串,两个父代个体的交换点选择相同,因此子代个体完全一样.这一事实将加速群体过早趋于同一化,从而导致进化进程的早熟现象.然而在遗传规划中,当两个相同的个体进行交叉时,所产生的子代个体一般不相同,除非两个子代个体各自的交叉点恰好相同,但这种情况发生的概率较小.在遗传算法和遗传规划中,选择操作都使群体趋于同一化.但在遗传规划中,交叉操作将施加一个偏离同一化趋势的反平衡作用.因此,在遗传规划中由于交叉操作的作用,同一化的发生是不可能的.

在遗传规划中,除了选择和交叉这两个主要的操作外,还有五个辅助的遗传操作:变异、排列、编辑、封装、自残.由于这些操作仅是根据需要偶然使用,所以不再逐一介绍.

15.2.5 结果标定

结果标定的第一种方法是找出一个在整个进化过程中的全局最佳个体.在进化中的每一代群体中都有最佳个体存在.在每一代进化完后,若当前代的最佳个体优于前一代,则将其存入计算机缓冲区,取代前一代的最佳个体;否则仍保留前一代的最佳个体.当终止准则满足,进化停止时,计算机缓冲区中的个体就作为整个进化进程的全局最佳个体.

第二种方法是在进化停止时标定出群体中的最佳个体，即取最后一代的最佳个体作为全局最佳个体.

15.2.6　控制参数

在遗传规划中，共有十几个控制参数，下面只介绍其中常用的 5 个参数. 参数值的选取一般要依赖于求解的问题，这里不具体讨论它们的最优选取，仅提供一组值供参考.

(1)群体规模 N，可取 $N=500$；

(2) 最大进化代数 G，可取 $G=51$，包括初始代和后继的 50 代；

(3)交叉概率 P_c 可取 $P_c=0.90$；

(4)个体算法树最大深度 $D1$，可取 $D1=17$，即在进化进程中由交叉产生的树的最大深度不超过 17；

(5)初始代个体算法树最大深度 $D2$，可取 $D2=6$，即在初始群体中随机产生的算法树的最大深度不超过 6.

§15.3　遗传规划在符号回归中的应用

符号回归(Symbolic Regression)问题主要是确定用符号表示的函数，这种函数能够以给定的精度拟合给定的数据. 这一工作类似于参数回归，不同之处在于参数回归要求事先给定具体函数形式，而符号回归则不需要事先给定具体函数形式. 因此，符号回归在工程中具有更广泛的应用价值.

遗传规划在符号回归中的应用是利用个体理想值与实际值之间的误差作为遗传的驱动力，这种进化模式称为误差驱动进化. 一般情况下，符号回归的进程包括数学函数形态的确定，函数中数字常量和系数的确定.

下面以降雨量状况预测问题为例，具体说明遗传规划在符号回归中的应用.

某地 10 月份降雨量与预报因子的实测数据如表 15.1 所示. 要解决的问题是确定该地降雨量(用 y 表示)与预报因子(用 x 表示)之间的关系，以便于对以后的降雨量状况做出预测.

表 15.1　某地 10 月份降雨量与预报因子之间的关系

年　号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
预报因子	97	47	95	84	96	113	24	16	47	37	42	14	23	35	27	85	96
实测值	34	17	28	25	31	52	8	6	14	9	13	3	14	10	9	36	48

表 15.1 中的 17 对实测数据构成了适应度计算事例，取端点集为

$$T=\{x,\rho\}$$

其中 x 为变量，ρ 为临时随机常数. 在随机生成初始个体时，只要临时随机常数 ρ 被选作为树的一个叶子，便在特定区间内生成具有特定数据类型的随机数，并代替临时随机常数 ρ. 对于本例，随机常数 ρ 的数据类型应为浮点型，取值区间可定义在[−20 ，+20]范围之内.

函数集中应将所有能估计到的反映自变量和因变量之间本质联系的函数搜集起来. 本例的函数集定义为

$$F=\{+,*,\exp,\log,\uparrow\}$$

群体规模取为 $n=4$，初始群体是由变量 x，常量 ρ 和函数+、*、exp、log、↑随机构成的复

杂数学函数. 设第 0 代群体为:

个体 1:　$-4.3+1.21*x$

个体 2:　$0.667 * \exp(0.071*x)$

个体 3:　$-12.72+1.77*\log(x)$

个体 4:　$1.242 *x \uparrow 0.76$

个体在表 15.1 的实测数据中运行,个体原始适应度定义为

$$r(i,t)=\sum_{j=1}^{17}|s(i,j)-y(j)|$$

其中 $s(i,j)$ 为个体 i 在计算事例 j 下的返回值, $y(j)$ 为计算事例 j 的实测值. 在本例中,标准适应度取为原始适应度.

基于表 15.1 的适应度计算事例,各计算机程序(个体)的返回值及其与实测值的误差绝对值之和如表 15.2 所示.

表 15.2　各计算机程序的返回值及其与实测值的误差绝对值总和

年　号	预报因子	实测值	计算程序返回值 $s(i,j)$			
	x	y	个体 1	个体 2	个体 3	个体 4
1	97	34	113.070	653.299	−4.634	40.185
2	47	17	52.570	18.766	−5.915	23.169
⋮	⋮	⋮	⋮	⋮	⋮	⋮
17	96	48	111.860	608.532	−4.652	39.870
误差绝对值总和 (适应度)			753.280	5123.380	457.493	118.518

由表 15.2 可见,第 0 代群体平均适应度为 1613.168,个体 4 的适应性最优.

一般情况下,在随机生成的初始群体中,大量个体具有较差的适应度,如本例中的个体 2 和个体 1. 但是,仍然会有一些适应度较好的个体,如本例中的个体 4 和个体 3. 这种适应度差别将在以后的遗传进化中得到改善.

下面根据适应度-比例原则进行选择和交叉操作. 在选择操作中,个体 2 被淘汰,个体 4 被复制 2 份. 于是,选择操作后的第一代子群体变为:

个体 1:　$-4.3+1.21*x$

个体 2:　$-12.72+1.77*\log(x)$

个体 3:　$1.242* x\uparrow 0.76$

个体 4:　$1.242* x\uparrow 0.76$

交叉操作仍根据适应度-比例原则进行,即个体适应性越好,进行交叉的概率越大. 假定进行交叉的个体为 2 与 3 以及 1 与 4,交叉后的第一代子群体为:

个体 1:　$-4.3+1.21*x\uparrow 0.76$

个体 2:　$-12.72+1.77* x\uparrow 0.76$

个体 3:　$1.242 * \log(x)$

个体 4:　$1.242 * x$

直观上看,如果父本个体在解决问题时较为有效的话,那么它们的某些部分很可能有重要价值. 通过这些有价值部分的随机组合,就很可能获得具有更好适应性的新生个体. 当某代的群体完成选择和交叉操作后,新生群体就取代了旧群体. 然后,利用一组计算事例,重新对新生代群体中的个体进行适应性评价. 这一进程可不断重复. 经过若干代后,个体的平均适应性不

断提高，而且，这些个体能快速、有效地适应环境的变化.

对于本例，基于表 15.1 的适应度计算事例，第 1 代群体各计算机程序的返回值及其与实测值的误差绝对值之和如表 15.3 所示.

表 15.3 各计算机程序的返回值及其与实测值的误差绝对值总和

年 号	预报因子 x	实测值 y	计算程序返回值 $s(i,j)$ 个体 1	个体 2	个体 3	个体 4
1	97	34	34.850	44.471	5.682	120.474
2	47	17	18.273	20.254	4.782P	58.374
⋮	⋮	⋮	⋮	⋮	⋮	⋮
17	96	48	34.543	44.022	5.669	119.232
误差绝对值总和（适应度）			75.314	98.908	276.225	857.676

由表中可见，有些个体适应性改善很大，如个体 2 的适应度从 457.493 改进为 98.908. 第 1 代群体平均适应度 327.031 与第 0 代群体平均适应度 1613.168 相比，也大为改善.

上述进化进程可不断重复，直到取得满意结果.

遗传规划在符号回归中的应用范围较广，它可以涉及数学恒等式的确定、科学定律的实验发现、符号积分、符号微分、符号微分方程求解、符号积分方程求解、一般函数方程求解、求函数的逆函数、工程与经济问题中的建模和预测，以及计算机图像压缩等诸多方面. 此外，遗传规划还可用于布尔复合算子、人工蚂蚁问题以及最优控制等领域.

最后，将遗传规划求解问题时的重要特征归纳如下：

产生的结果具有层次化的特点；

无需事先确定或限制最终答案的结构或大小，遗传规划将根据环境自适应地确定问题解的结构和大小；

输入、中间结果和输出是问题的自然描述，无需或很少需要对输入数据的预处理和对输出结果的后处理. 进化过程中所产生的计算机程序由对于问题自然描述的函数组成.

必须指出，在遗传规划中，个体结构的变化是主动的，它们并不是对问题答案的被动式编码，这一点完全不同于遗传算法.

§15.4 进化策略与进化规划

15.4.1 进化策略

进化策略(Evolution Strategies，ES)的研究始于 1964 年，由德国学者 Rechenberg I 和 Schwefel H P 在 Technical University of Berlin 最先提出. 当初主要是用于试验处理流体动力学问题，如弯管形状优化(具有最小能量损失的柔性弯管的形状). 这是一种求解数值优化问题的算法.

Rechenberg 提出了最原始的进化策略，称为(1+1)－ES，并发展了收敛速度理论. (1+1)－ES 是一个简单的变异选择机制，每代通过高斯变异作用在一个父本个体上来产生一个子代个体. Rechenberg 还提出了$(\mu+1)$－ES，这个策略是将 $\mu(\mu>1)$ 个个体经过选择形成的一个子代个体替换掉最差的一个父代个体. Schwefel 在此基础上提出了$(\mu+\lambda)$－ES 和(μ,λ)－ES，它们都是针对多父本和多子代的，在算法过程中，μ 个父本产生 $\lambda(\lambda>1)$ 个子代. $(\mu+\lambda)$－ES 表示在算法中这$(\mu+\lambda)$个候选解同时参加竞争，以选择产生新一代(一般生存数目为 μ).

$(\mu,\lambda)-ES$ 表示在算法中只有这 λ 个子代个体作为参加竞争的候选解，而父本被完全取代. 可以称 $(\mu+\lambda)-ES$ 为"父子混合选择"的进化策略，而 $(\mu,\lambda)-ES$ 为"自然选择". 这两种策略(尤其是后者)在进化策略中占有重要地位.

下面介绍利用 $(\mu,\lambda)-ES$ 求解优化问题(P)

$$\max\{f(\underline{x}):\underline{x}\in D\subset \mathbf{R}^n\}\quad f:D\subset \mathbf{R}^n\to \mathbf{R}^1$$

的步骤.

1. 初始化

确定群体规模 N 与终止进化准则，在 D 中随机选择 N 个元(向量) $\underline{x}_i(0)(i=1,2,\cdots,N)$ 以组成初始群体 $\boldsymbol{X}(0)=\{\underline{x}_1(0),\underline{x}_2(0)\},\cdots,\underline{x}_N(0)\}$，置 $k=0$.

2. 产生中间群体

执行 M 步 $(M\geqslant N)$ 如下操作.

① 以等概率从 $\boldsymbol{X}(k)$ 中选择用作父本的个体 $\underline{x}_{i_1}(k),\underline{x}_{i_2}(k)$.

② 以交叉算子作用于 $\underline{x}_{i_1}(k),\underline{x}_{i_2}(k)$ 以产生中间个体 S_i'.

③ 以变异算子作用于 S_i' 以产生中间个体 S_i.

3. 选择

按适应度值 $F(S_i)(1\leqslant i\leqslant M)$ 的大小从中间群体 $\{S_1,S_2,\cdots,S_M\}$ 中选取 N 个个体以组成下一代群体 $\boldsymbol{X}(k+1)$.

4. 终止检验

判断 $\boldsymbol{X}(k+1)$ 是否满足终止进化条件，如满足，则停止进化，否则置 $k=k+1$. 转步 2.

从形式上看，进化策略和遗传算法十分类似，但进化策略不使用优化变量的染色体编码，而是直接作用于实变量. 另外，在交叉算子与变异算子的构造上，进化策略与遗传算法是有区别的. 进化策略使用的交叉算子是以两个个体生成一个个体的操作.

常见的交叉算子有"中间交叉"(intermediate crossover)以及"随机交叉"(random crossover)等. 对于给定的两个父本向量 $\underline{x}_{i_1}$ 和 $\underline{x}_{i_2}$，中间交叉产生所给父本向量的算术平均：

$$\underline{S_i'}=\frac{1}{2}(\underline{x}_{i_1}+\underline{x}_{i_2})$$

随机交叉定义为从所给父本向量中的随机抽取：

$$\underline{S_i'}=\mathrm{Random}(\underline{x}_{i_1}+\underline{x}_{i_2})$$

交叉算于也可以取如下加权平均交叉的形式：

$$\underline{S_i'}=(\underline{x}_{i_1}-\theta(\underline{x}_{i_2}-\underline{x}_{i_1})$$

其中 $\theta\in[0,1]$ 为一致随机变量. 若 θ 仅取随机数 0 或 1，则即为随机交叉；若 θ 取定值 $\frac{1}{2}$，则即为中间交叉.

不同于遗传算法的逐点变异，进化策略中的变异算子通常是通过对所作用的个体施加某个随机扰动来实现的. 例如，对 S_i' 的变异可定义为

$$\underline{S_i}=\underline{S_i'}+N(0,\underline{\boldsymbol{\sigma}_i})$$

其中 $N(0,\underline{\boldsymbol{\sigma}_i}$ 表示均值为 0，偏差为 $\underline{\boldsymbol{\sigma}_i}$ 的独立正态分布随机向量. 由于进化策略中的变异从一定意义上可理解为一种爬山搜索，$\underline{\boldsymbol{\sigma}_i}$ 也常称作为步长.

采用定步长控制策略，使算法有结构简单、易于实施的优点，但算法常常收敛缓慢、易于停滞于

局部极小. Schwefel 发展了一类参数自校正策略. 在这个策略中，群体中的个体不再仅由问题(P)的单个可行解描述，而是包括了可行解向量以及相应扰动向量的向量对. 例如，第 k 代群体中的个体可描述为$(\underline{X}_i(k),\underline{\sigma}_i(k))(i=1,2,\cdots,N)$. 在这个向量对中，扰动向量不仅提供了变异自身，而且也提供了对可行解向量所变异的结果. 设$(\underline{S}',\underline{\sigma}')$为某一给定个体，依 Schwefel 的定义，对这一个体的变异产生的新个体$(\underline{S},\underline{\sigma})$为

$$\begin{cases}\underline{\sigma}=\underline{\sigma}'\exp(\underline{N}(0,\Delta\underline{\sigma}'))\\ \underline{S}=\underline{S}'+\underline{N}(0,\Delta\underline{\sigma}')\end{cases}$$

其中 $\Delta\underline{\sigma}$ 为步长. 理论分析与数值试验均表明，采用 Schwefel 的参数自校正策略可显著改善进化策略算法的计算效率与优化性能.

对于进化策略收敛性及收敛速度的分析研究，虽然已有一些结果，例如，现已证明了(1+1)－ES 的渐进收敛性，也有了一种特殊定义的收敛速度的估计，但总的来说，还缺乏具有普遍意义的工作，特别是，有关$(\mu+\lambda)$－ES，(μ,λ)－ES 及 Schwefel 的参数自校正进化策略的一般收敛性尚未得到严格的证明.

15.4.2 进化规划

进化规划(Evolutionary Programming, EP)是 20 世纪 60 年代由美国的 Fogel L J 等人提出的一类进化算法，其目的在于求解静态或非静态时间序列预测问题. 进化规划的设计思想类似于进化策略，即主要利用变异与选择机制对系统进行优化，但进化规划往往用于处理更复杂的系统优化问题，如人工智能领域的优化问题等.

求解问题(P)的进化规划可描述如下：

1. 初始化

确定群体规模 N 与终止进化准则，在 D 中随机选取 $2N$ 个个体以组成初始群体$\underline{x}_i(0)(i=1,2,\cdots,2N)$，即：

$$X(0)=\{\underline{x}_1(0),\cdots,\underline{x}_N(0),\underline{x}_{N+1}(0),\cdots,\underline{x}_{2N}(0)\}$$

置 $k=0$.

2. 产生新一代群体

① 计算当前群体中个体的适应度值 $F(\underline{x}_i(k)),1\leqslant i\leqslant 2N$.

② 依据某种竞争规则，选择$\{\underline{x}_i(k),i=1,2,\cdots,2N\}$中获胜概率最高的 N 个个体作为父本. 不妨设这 N 个父本个体为

$$\{\underline{x}_1(k),\cdots,\underline{x}_N(k)\}$$

并令

$$x_i(k)=(x_{i1}(k),x_{i2}(k),\cdots,x_{in}(k))\quad(i=1,2,\cdots,N)$$

③通过变异算子作用于父本$\underline{x}_i(k)(1\leqslant i\leqslant N)$以产生新个体

$$\underline{S}_i=(S_{i1},S_{i2},\cdots,S_{in})\quad(1\leqslant i\leqslant N)$$

其中 $S_{ij}=x_{ij}(k)+N(0,\alpha_{ij}F(x_{ij}(k))+\beta_{ij})(j=1,2,\cdots,n)$，其中 $N(\cdot,\cdot)$是正态分布随机变量，α_{ij},β_{ij} 是可调实参数.

④取新一代群体 $X(k+1)=\{\underline{x}_1(k),\cdots,\underline{x}_N(k),\underline{S}_1,\cdots,\underline{S}_N\}$.

3. 终止检验

如果 $X(k+1)$满足终止进化条件,则停止并输出最优解,否则置 $k=k+1$,转步骤 2.

在选择父本的过程中,竞争规则和获胜率通常按如下定义:

任一个体$\underline{x}_i(k)\in X(k)$称为在一次竞争中获胜,如果随机选取一个体$\underline{x}_j(k)\in X(k)$有 $F(\underline{x}_i(k))\geqslant F(\underline{x}_j(k))$. 在规定的 m 次竞争中,如果$\underline{x}_i(k)$有 α 次获胜,则比值$\frac{\alpha}{m}$称为个体$\underline{x}_i(k)$的获胜率.

进化规划只利用了变异算子的作用,而没有利用交叉算子.

15.4.3 遗传算法与进化策略和进化规划的比较

作为进化计算的三大典型执行策略,GA,ES 与 EP 均以作用于生物种群,模拟生物进化机制为基础,且其发展有逐渐融合的趋势,但注意到这三类不同执行策略之间的差别是有益的.

(1)GA 的基本作用对象是优化变量的染色体编码,从而它是通过求解组合优化问题(P)求解

$$\max\{\ F(S):S\in E\ \}$$

其中 $E=\{0,1\}^L$ 为问题(P)中可行解空间 D 的编码空间,用来求解优化问题(P). 而 ES 与 EP 直接作用于表达个体的实变置本身,其算法是在问题(P)的可行解空间中直接进行搜索.

(2) GA 的仿生基点在于通过遗传算子的作用改变个体的基因结构,从而强调交叉作用方式与繁殖后代的遗传手段. ES 与 EP 注重父代与子代之间的性状联系,从而强调无论遗传过程怎样发生,父代与子代之间必须保持服从概率分布的性状.

(3) ES 的性状联系表现在用作父本的个体与子代个体之间,而 EP 的性状表现于父代种群与子代种群的群体之间. 因此前者可引入交叉操作,而后者则不宜.

(4) 变异在 GA 中只是提供等位基因的一个恢复机会,不是重要的搜索算子,而在 ES 与 EP 中,变异是最基本的搜索.

(5) 在选择机制上,ES 利用等概率方式(与个体适应性无关)产生用于繁殖的父本,而 GA 和 EP 强调选择对于个体适应性的依赖与概率机制.

目前,进化计算已被广泛认为是可成功应用于计算机科学、工程技术、管理科学和社会科学等领域的一种自组织、自适应人工智能技术. 特别是,它对于求解复杂优化问题所显示出的巨大潜力愈来愈引起众多学科的研究者的兴趣.

世界之奇妙,万物之和谐,存在于人类进化自身的优化原理或许正是求解任何复杂优化问题的天赐之道.

参 考 文 献

[1] Holland J H. Adaptation in Natural and Artificial Systems. Ann Arbor, MI: University of Michigan Press, 1975.

[2] Goldberg D E. Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley, 1989.

[3] Fogel D B. An introduction to simulated evolutionary optimization. IEEE Transactions on neural networks, 1994, Vol. 5, No. 1, pp. 3～14

[4] Rudolph G. Convergence analysis of canonical genetic algorithms. IEEE Transactions on neural networks, 1994, Vol. 5, No. 1, pp. 96～101

[5] Qi XF, Palmieri F. Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space, I. Basic properties of selection and mutation. IEEE Transactions on neural networks, 1994, Vol. 5, No. 1, pp. 102～119

[6] Qi XF, Palmieri F. Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space, II. Analysis of the diversification of crossover. IEEE Transactions on neural networks, 1994, Vol. 5, No. 1, pp. 120～129

[7] 陈国良，王煦法，庄镇泉，王东生. 遗传算法及其应用. 北京：人民邮电出版社，1996.

[8] 云庆夏，黄广球，王战权. 遗传算法和遗传规划——一种搜索寻优技术. 北京：冶金工业出版社，1997.

[9] 刘勇，康立山，陈毓屏. 非数值并行计算——遗传算法. 北京：科学出版社，1995.

[10] 徐宗本，李国. 解全局优化问题的仿生类算法(I)—— 模拟进化算法. 运筹学杂志，1995，Vol. 14，No. 2，pp. 1～13.

[11] 王艳丰，王众托. 关于遗传算法图式定理的分析研究. 控制与决策，1996，Vol. 11，Suppl. 1，pp. 221～224

[12] 梁艳春. 遗传算法与人工神经网络若干问题研究：[博士学位论文]. 长春：吉林大学，1997.

[13] 周春光，周国芹，李博，程彦丰，梁艳春. 一种快速收敛的遗传算法. 软件学报，1996，Vol. 7，Supplement，pp. 311～314

[14] 周春光，周国芹，程彦丰，常迪，梁艳春. 一种克服遗传算法收敛于局部极小的方法. 小型微型计算机系统，1997，Vol. 18，No. 3，pp. 46～49

[15] 周春光，梁艳春，田勇，胡成全，孙新. 基于模糊神经网络味觉信号识别的研究. 计算机研究与发展，1999，Vol. 36，No. 4，pp401～409

[16] 梁艳春，周春光，王在申. 人工神经网络应用地下洞室围岩参数识别的研究. 模式识别与人工智能，1996，Vol. 9，No. 1，pp71～77

[17] 梁艳春，冯大鹏，周春光. 遗传算法求解旅行商问题时的基因片段保序. 系统工程理论与实践，2000，Vol. 20，No. 4，pp. 7～12

[18] 梁艳春，周春光，李寿范. 基于遗传算法的 Rosenbrock 函数优化问题的研究. 软件学报，

1997, Vol. 8, No. 9, pp. 701～708

[19] 梁艳春,周春光,王在申. 基于扩展串的等效遗传算法的收敛性. 计算机学报,1997, Vol. 20, No. 8, pp. 686～694

[20] 梁艳春,王在申,周春光. 选择和变异操作下遗传算法的收敛性研究. 计算机研究与发展, 1998, Vol. 35, No. 7, pp655－660

[21] 梁艳春,周春光. 基于神经网络方法的包装件非线性特性识别的研究. 力学学报, 1997, Vol. 29, No. 4, pp. 496～500

[22] Zhou Chunguang, Zhou Guoqin, Liang Yanchun, Wang Yan. Backtracking Genetic Algorithms. Chinese Journal of Advanced Software Research, 1997, Vol. 4, No. 3, pp. 255～265

[23] Liang Yanchun, Zhou Chunguang. Advances in identification of nonlinear characteristics of packaging based on computational intelligence. Mechanics Research Communications, 2000, Vol. 27, No. 1, pp. 15～20

[24] Liang Yanchun, Gong Wenying, Yang Xiaowei, Zhou Chunguang. Identifi- cation of nonlinear characteristics of packaging based on genetic evolutionary neural networks. Mechanics Research Communications, 1998, Vol. 25, No. 4, pp. 395～403

[25] Liang Yanchun, Zhou Chunguang, Wang Zaishen. Identification of restoring forces in non—linear vihration systems hased on neural networks. Journal of Sound and Vibration, 1997, Vol. 206, No. 1, pp. 103～108

[26] Liang Yanchun, Wang Zheng, Yang Xiaowei, Zhou Chunguang. Identification of non-linearities in cushioning packaging using neural networks with fuzzy adaptive control. Mechanics Research Communications, 1997, Vol. 24, No. 4, pp. 447～455

[27] Liang Yanchun, Yang Xiaowei, Zhou Chunguang, Wang Zaishen. Application of neural networks to idetitification of nonlinear characteristics in cushioning packaging. Mechanics Research Communications, 1996, Vol. 23, No. 6, pp. 607～613

[28] 田夫汉,周春光,田力汉. 遗传算法中基因缺失的预防. 小型微型计算机系统, 2000, Vol. 21, No. 9, pp947～949

[29] 周春光,周国芹,梁艳春. 遗传算法中的重组操作. 吉林大学自然科学学报, 1996, No. 1, pp. 21～24

[30] 周国芹,周春光,梁艳春. 回溯遗传算法. 吉林大学自然科学学报, 1996, No. 4, pp. 37～42

[31] Zhou Guoqin, Zhou Chunguang, Li Bo, Liang Yanchun. A Fast Convergent Genetic Algorithm. Beijing China: International Conference on Neural Information Processing (ICONIP'95). oct 1995, Volumel of 2, pp. 271～274

[32] Zhou Guoqin, Zhou Chunguang, Li Bo, Liang Yanchun. Meng Zhaolong. A Method Avoiding Local Optima in Genetic Algorithms. Beijing China : International Conference on Neural Information Processing (ICONIP'95). oct. 1995, Volume 1 of 2, pp. 275～278